Course Learning Objectives

As a result of completing this course, you will be able to:

- Overview of Pentaho Data Integration Components
- Navigate the Spoon interface
- Create, preview, and RUN basic Transformations containing Steps and Hops
- Explore the Data Pipeline with persistent, drilldown and geo capabilities

- Transform some of the common flat file datasources, into reporting datasets:
    - Text, CSV Files
    - Excel
    - XML
    - JSON
- Perform the 5 basic database operations on a database table:
    - Create
    - Read
    - Update
    - Delete
    - Insert
- Implement strategies to track changes in dimension attributes to report historical data (slowly changing dimensions – Technical or Surrogate Keys)

- Enrich the dataset

-

Useful Links

For information about instructor-led training on the topics covered in this guide, visit

Pentaho Training:
`http://www.pentaho.com/service/training`

KETTLE Forum:
`http://forums.pentaho.com/forumdisplay.php?135-Pentaho-Data-Integration-Kettle`

Pentaho Knowledge Base:
`https://support.pentaho.com/hc/en-us`

Pentaho Documentation:
`https://help.pentaho.com/Documentation/`

Pentaho Wiki:
`http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+Steps`

Pentaho Data Integration Development Guidelines:
`https://support.pentaho.com/hc/en-us/articles/205715016-Best-Practice-Pentaho-Data-Integration-Development-Guidelines`

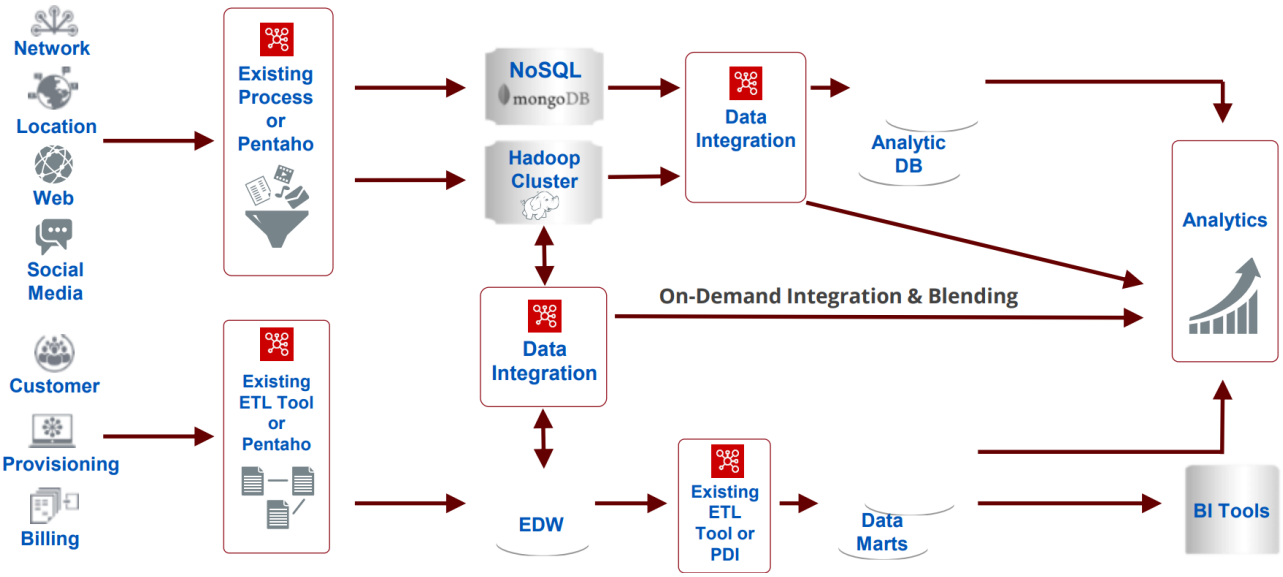
Pentaho Data Integration Design Guidelines:
`https://support.pentaho.com/hc/en-us/articles/205715026-Best-Practices-PDI-Design-Guidelines`

1: Getting Started with Pentaho Data Integration

Topics covered in this section:

- Pentaho Data Integration platform
- Navigating Spoon Interface
- KETTLE Configuration Files
- Adding JDBC Driver
- Repository

Pentaho Data Integration (PDI) is a powerful extract, transform, and load (ETL) solution that uses an innovative metadata-driven approach. It includes an easy to use, graphical design environment for building ETL jobs and transformations, resulting in faster development, lower maintenance costs, interactive debugging, and simplified deployment.



Pentaho Data Integration is an extremely flexible tool that addresses a broad number of use cases including:

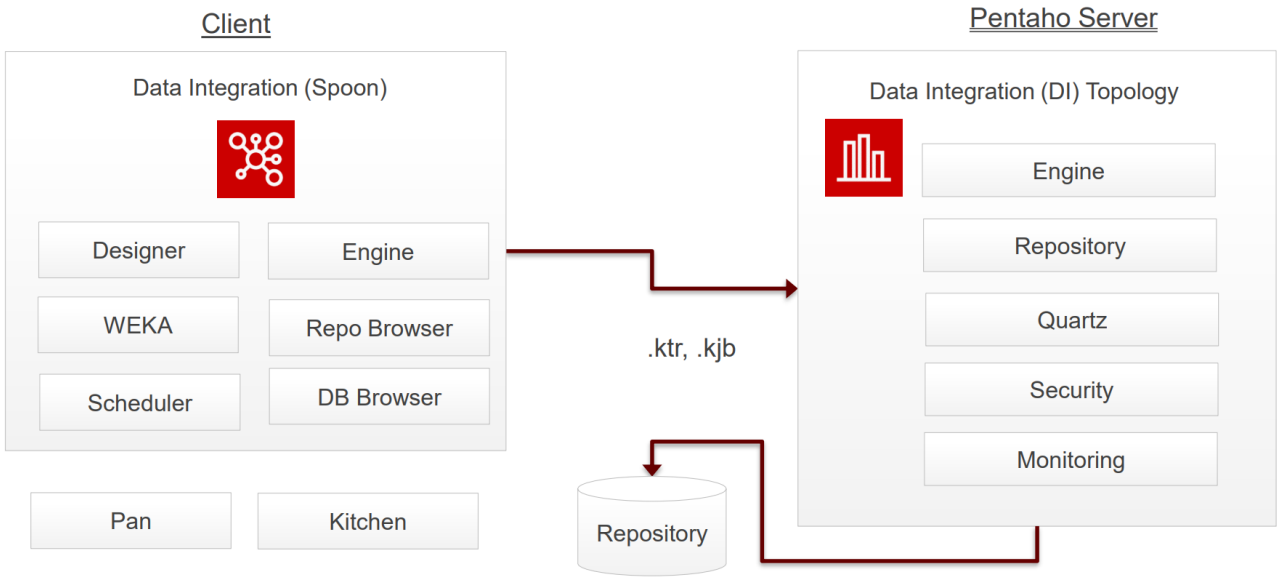
- Data warehouse population with built-in support for slowly changing dimensions and surrogate key creation
- Data migration between different databases and applications
- Loading huge data sets into databases taking full advantage of cloud, clustered and massively parallel processing environments
- Data Cleansing with steps ranging from very simple to very complex transformations
- Data Integration including the ability to leverage real-time ETL as a data source for Pentaho Reporting
- Rapid prototyping of ROLAP schemas
- Hadoop functions: Hadoop job execution and scheduling, simple Hadoop map/reduce design, Amazon
- EMR integration

Pentaho Data Integration features and benefits include:

- Installs in minutes; you can be productive in one afternoon
- 100% Java with cross platform support for Windows, Linux and Macintosh
- Easy to use, graphical designer with over 100 out-of-the-box mapping objects including inputs,
- transforms, and outputs
- Simple plug-in architecture for adding your own custom extensions
- Enterprise Data Integration server providing security integration, scheduling, and robust content management including full revision history for jobs and transformations
- Integrated designer (Spoon) combining ETL with metadata modelling and data visualization, providing the perfect environment for rapidly developing new Business Intelligence solutions
- Streaming engine architecture provides the ability to work with extremely large data volumes
- Enterprise-class performance and scalability with a broad range of deployment options including dedicated, clustered, and/or cloud-based ETL servers

Pentaho Data Integration Architecture

The diagram below depicts the core components of Pentaho Server Enterprise Edition.



Spoon

Spoon is the design interface for building ETL jobs and transformations, and resides on your desktop. Spoon provides a drag and drop interface allowing you to graphically describe what you want to take place in your transformations which can then be executed locally within Spoon, on a dedicated Data Integration Server, or a cluster of servers.

Command Line Launchers: Kitchen & Pan

You can use PDI's command line tools to execute PDI content from outside of Spoon. Typically, you would use these tools in the context of creating a script or a Cron job to run the job or transformation based on some condition outside of the realm of Pentaho software.

- **Pan.** A standalone command line process that can be used to execute transformations and jobs you created in Spoon. The data transformation engine Pan reads data from and writes data to various data sources. Pan also allows you to manipulate data.
- **Kitchen.** A standalone command line process that can be used to execute jobs. The program that executes the jobs designed in the Spoon graphical interface, either in XML or in a database repository. Jobs are usually scheduled to run in batch mode at regular intervals.

Carte

Carte is a lightweight Web container that allows you to set up a dedicated, remote ETL server. This provides similar remote execution capabilities as the Data Integration Server, but does not provide scheduling, security integration, and a content management system.

Pentaho Server

The Pentaho Server hosts Pentaho-created and user-created content. It is a core component for executing data integration transformations and jobs using the Pentaho Data Integration (PDI) Engine. It allows you to manage users and roles (default security) or integrate security to your existing security provider such as LDAP or Active Directory.

The primary functions of the Pentaho Server are:

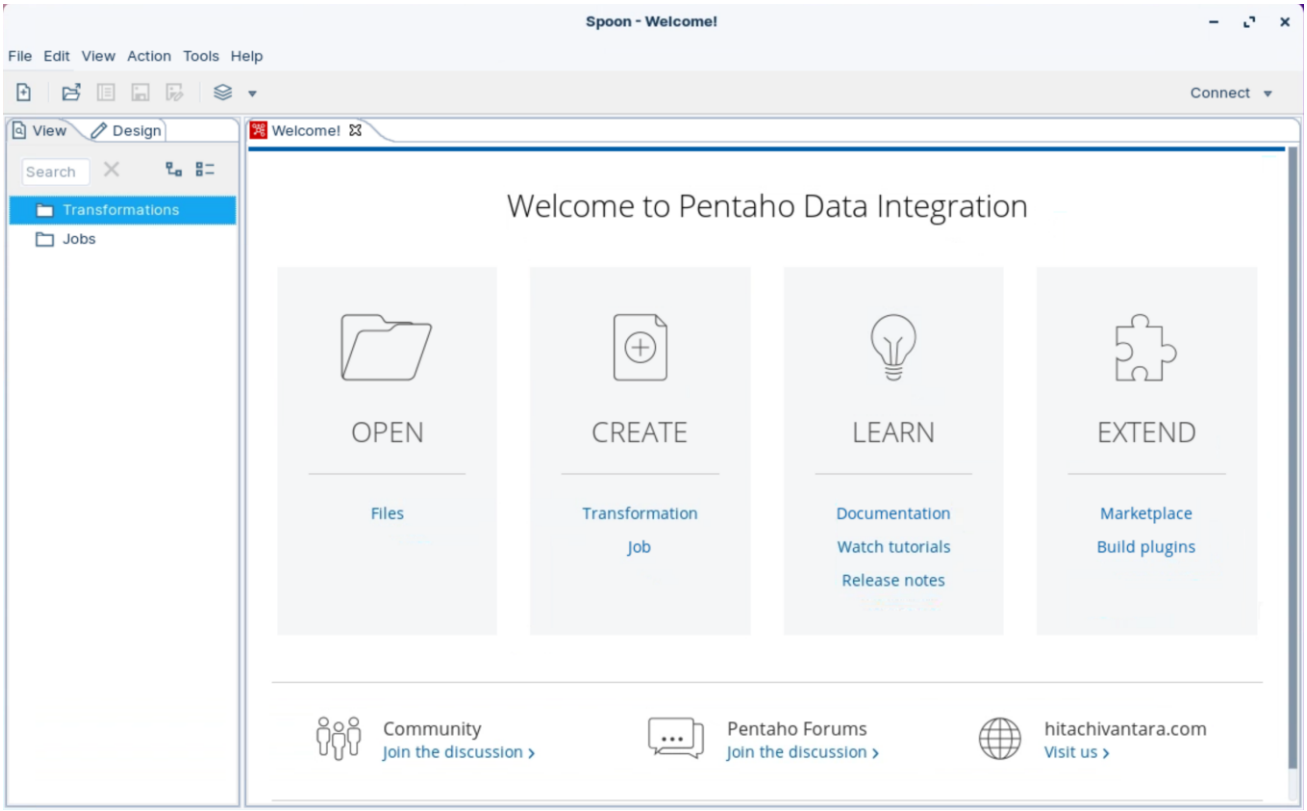
Execution	Executes ETL jobs and transformations using the Pentaho Data Integration engine
Security	Allows you to manage users and roles (default security) or integrate security to your existing security provider such as LDAP or Active Directory
Content Management	Provides a centralized repository that allows you to manage your ETL jobs and transformations. This includes full revision history on content and features such as sharing and locking for collaborative development environments.
Scheduling & Monitoring	Provides the services allowing you to schedule and monitor activities on the Data Integration Server from within the Spoon design environment (Quartz).

The Components Reference in Pentaho Documentation has a complete list of supported software and hardware.

Navigating the Spoon Interface

Spoon is a desktop application that you will use primarily as a graphical interface and editor for transformations and jobs. With Spoon, you can author, edit, run, and debug transformations and jobs. You can also use Spoon to enter license keys, add data connections, and define security.

The Welcome page contains useful links to documentation, community links for getting involved in the Pentaho Data Integration project, and links to blogs from some of the top contributors to the Pentaho Data Integration project.



There are a few different ways to start Spoon. The method that you should use depends on the way you installed Pentaho Data Integration (PDI).

Windows / Unix	Action
spoon.bat / spoon.sh	Starts Spoon
kichen.bat / kitchen.sh	Command Line for Jobs
pan.bat / pan.sh	Command Line for Transformations

Perspectives

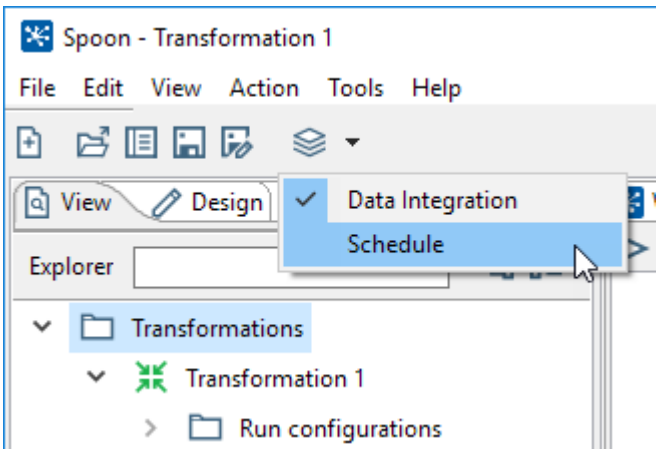
Pentaho Data Integration (PDI) empowers you with tools that include ETL and scheduling in one unified environment — the PDI client (Spoon) interface. This integrated environment enables you to work in close cooperation with business users to build business intelligence solutions more quickly and efficiently.

When you are working in the PDI client, you can *change perspectives* to easily switch back and forth from:

- Designing ETL jobs and transformations
- Scheduling jobs and transformations

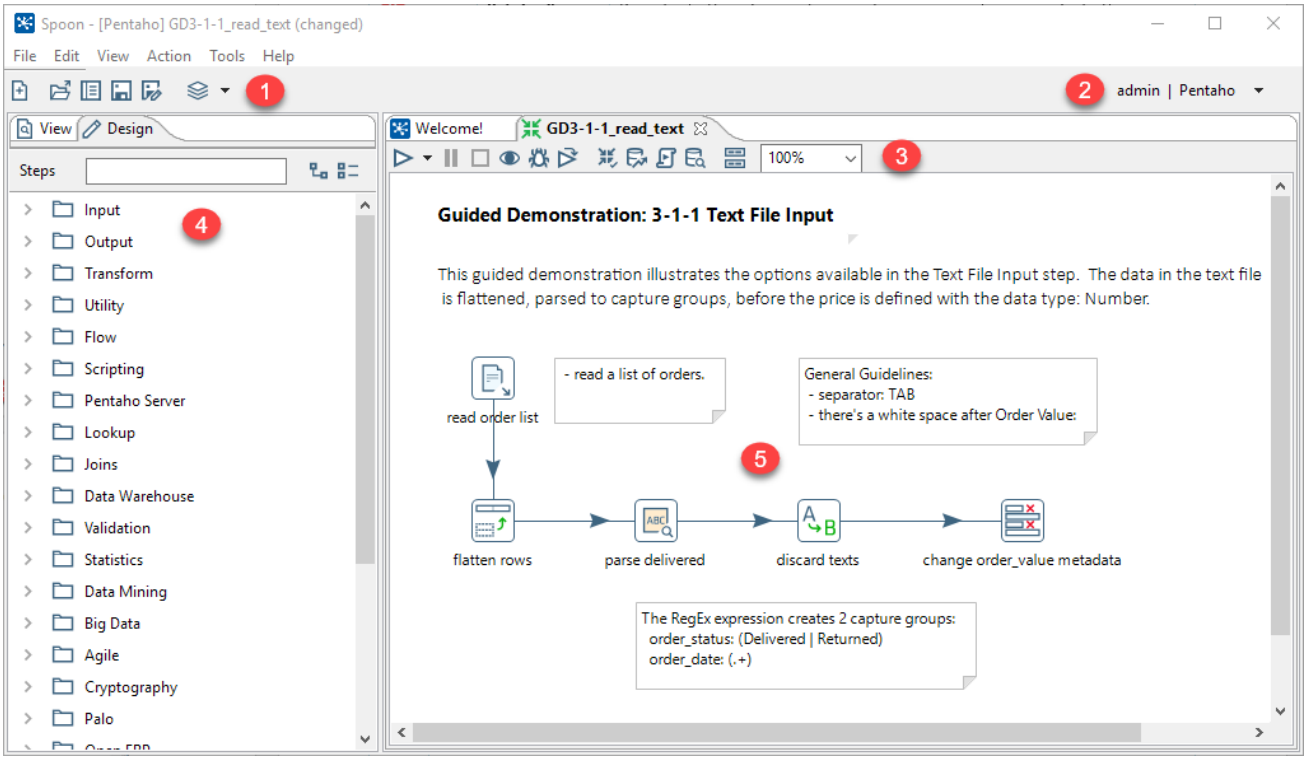
As users provide you with feedback about how the data is presented to them, you can also quickly make iterative changes to your data directly using our data inspection tools in the PDI client.

From within the PDI client, you can change perspectives using the Perspective icon in the toolbar.



Data Integration Perspective

The Data Integration perspective allows you to create transformations, jobs, and inspect your data allowing for iterative updates as you work.



Component	Name	Description
1	**Toolbar**	Single-click access to common actions such as create a new file, opening existing documents, save and save as. You can also toggle between the Data Integration and Schedule perspectives
2	**Repository**	Connect to the Repository
3	**Sub-toolbar**	Provides buttons for quick access to common actions specific to the transformation or job such as Run, Preview, and Debug.
4	**Design and View Tabs**	The **Design** tab of the Explore pane provides an organized list of transformation steps or job entries used to build transformations and jobs. Transformations are created by simply dragging transformation steps from the Design tab onto the canvas and connecting them with hops to describe the flow of data. The **View** tab of the Explore pane shows information for each job or transformation. This includes information such as available database connections and which steps and hops are used. In the image, the Design tab is selected.
5	**Canvas**	Main design area for building transformations and jobs.

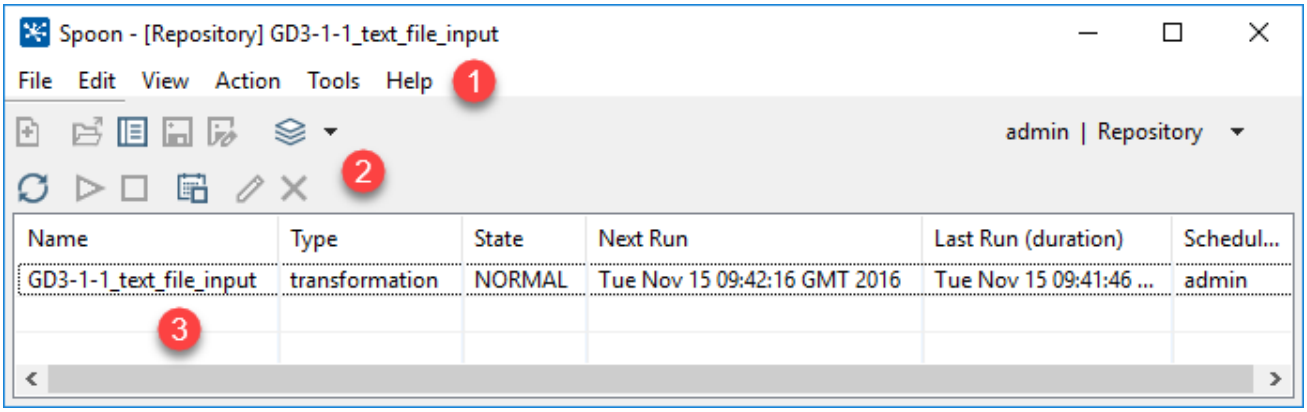
Icon	Description
	Create a new job or transformation
	Open transformation/job from file if you are not connected to a repository or from the repository if you are connected to one
	Explore the repository
	Save the transformation/job to a file or to the repository
	Save the transformation/job under a different name or file name (Save as)
	Run transformation/job and set run options; runs the current transformation from XML file or repository
	Pause transformation
	Stop transformation
	Preview transformation: runs the current transformation from memory. You can preview the rows that are produced by selected steps.
	Run the transformation in debug mode; allows you to troubleshoot execution errors
	Replay the processing of a transformation
	Verify transformation
	Run an impact analysis on the database
	Generate the SQL that is needed to run the loaded transformation.
	Launch the database explorer allowing you to preview data, run SQL queries, generate DDL and more
	Show execution results pane
	Lock transformation

Schedule Perspective

The Schedule perspective is used for managing schedules of jobs and transformations. When you have scheduled a job or transformation to occur at a specified time by selecting:

Action > Schedule

the resulting scheduled task appears in the listing of this perspective.



Component	Name	Description
1	**Menu bar**	The Menu bar provides access to common features such as properties, actions and tools. The right side of the menu bar is also where you can switch between perspectives.
2	**Main Toolbar**	The Main Toolbar provides single-click access to common actions such as edit, refresh, enable, disable, or delete.
3	**Schedule Panel**	Contains a list of schedules to select. Double-click on a highlighted schedule to access the **Schedule** dialog for editing.

Guided Demo: Configuring Spoon

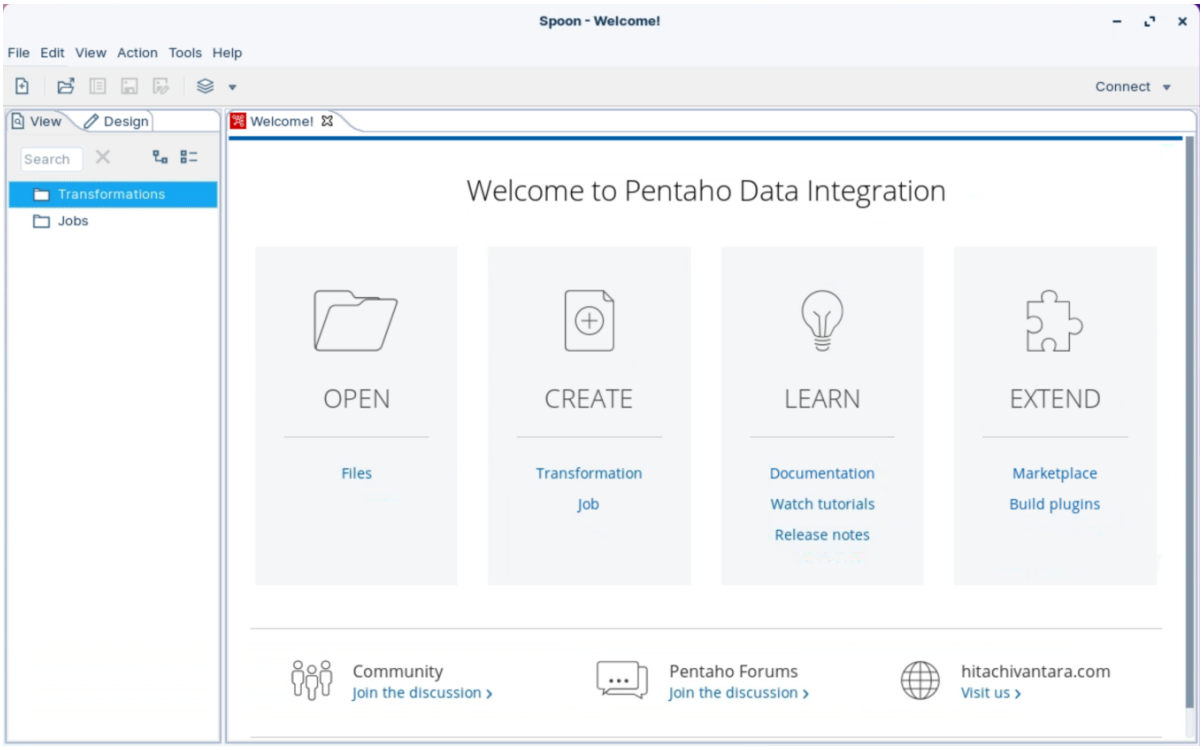
Introduction	This Guided Demo Introduces the student to Spoon features and customization options.
Objectives	In this guided demonstration, you will… • Launch Spoon, PDI's graphical designer interface. • Review the welcome page for information on Blogs, Books and Forums. • Open Spoon's 'Options' dialog. • Describe the common options, look & feel settings.

Launching Spoon

With an intuitive, graphical, drag and drop design environment and a proven, scalable, standards-based architecture, Data Integration is increasingly the choice for organizations over traditional, proprietary ETL or data integration tools.

1. To launch Spoon, PDI's graphical designer interface, from the Windows Start button:

   Start > All Programs > Pentaho Enterprise Edition > Design Tools > Data Integration



   Alternatively, you can start Spoon from the following path:

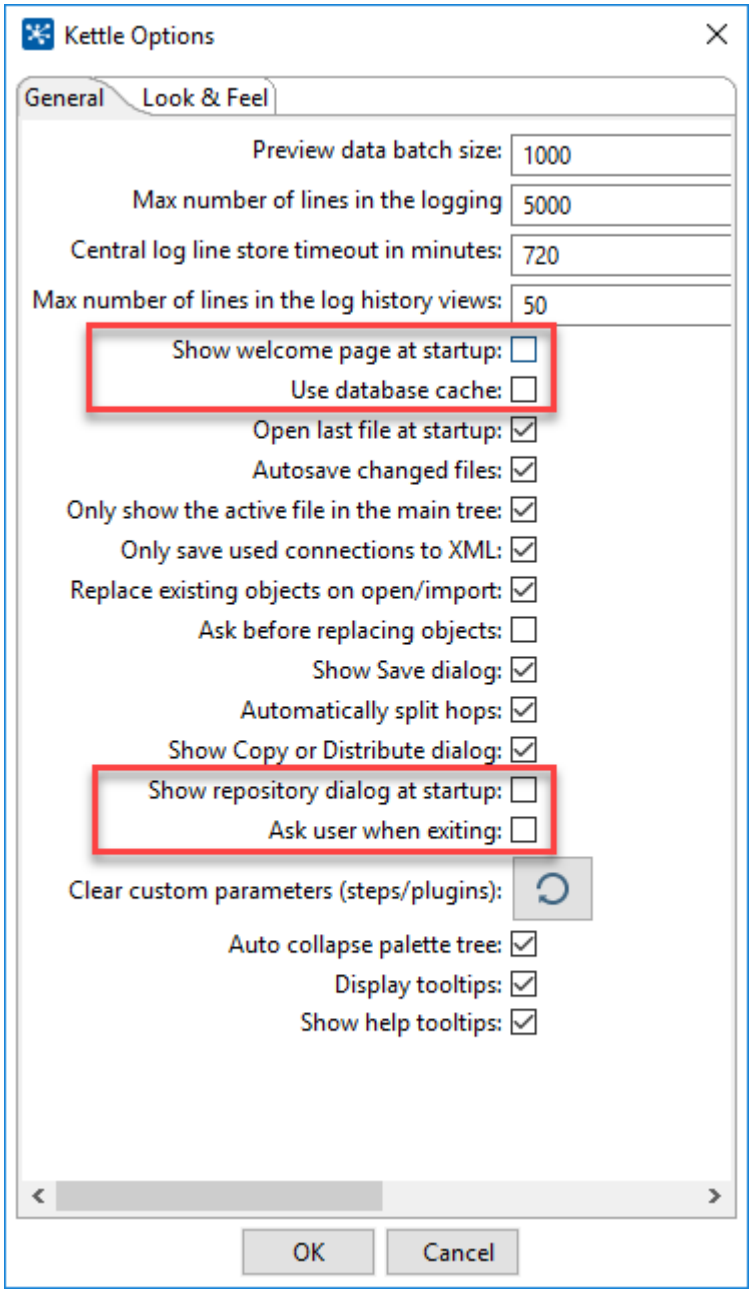
   [Path to Pentaho Installation]\design-tools\data-integration\spoon.bat

   [Path to Pentaho Installation]\design-tools\data-integration\spoon.sh

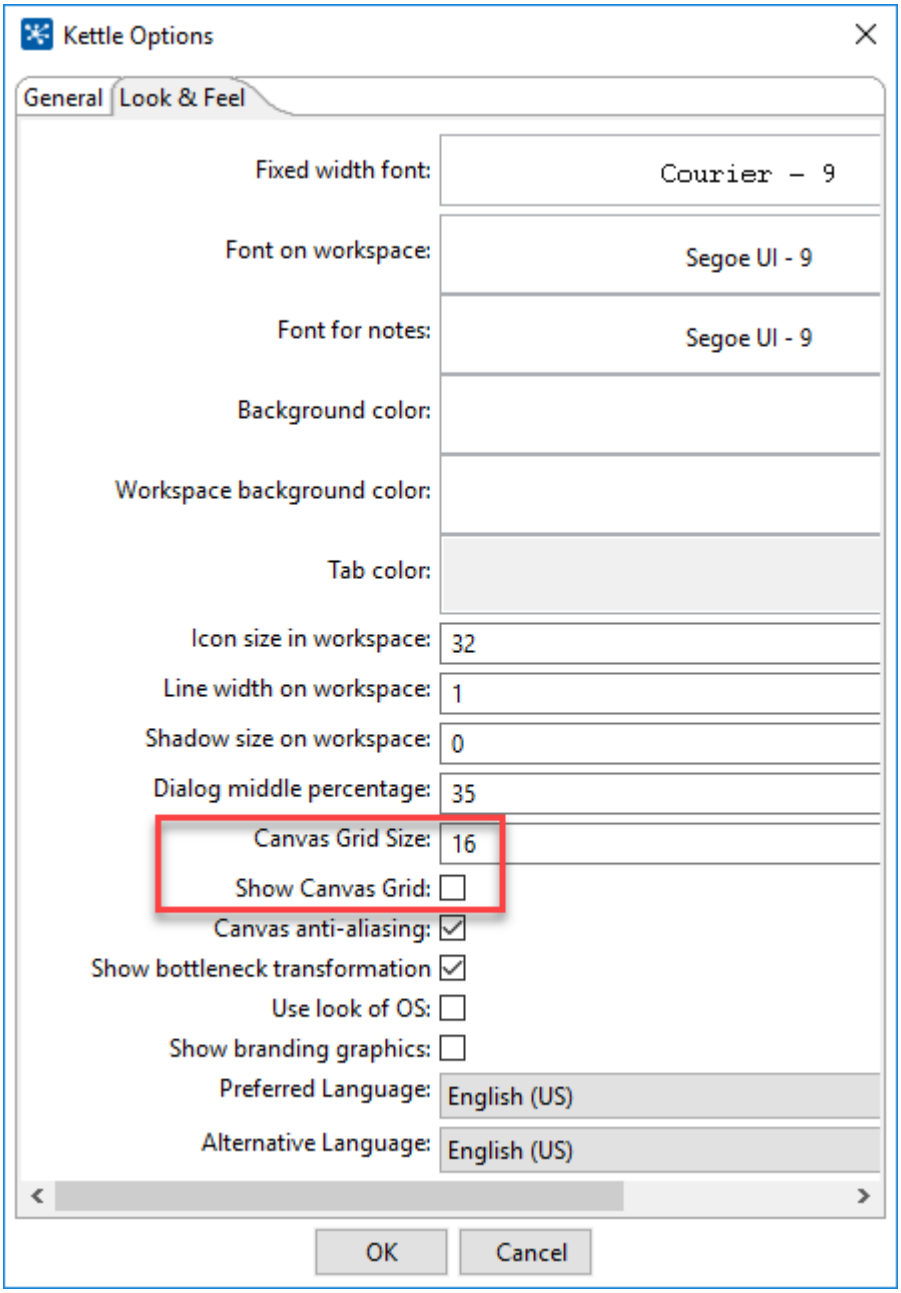
Modifying the Look and Feel of Spoon

To modify some of the graphical options:

1. In the menu bar, click Tools | Options.
2. In the General tab.
   - Uncheck the 'Show tips at startup?' checkbox.
   - Uncheck the 'Use database cache' checkbox.
   - Uncheck the 'Show repository dialog at startup?' checkbox.
   - Uncheck the 'Ask user when exiting' checkbox.



In the Look & Feel tab:



Feel free to use any font or grid size you like. However, 32 is the recommended setting for the grid size because it makes aligning steps on the canvas easy.

Although restarting Spoon after changing Look & Feel options is the best habit to have, not all options require a restart for the changes to take effect.

Pentaho Data Integration Configuration

The default Pentaho Data Integration (PDI) HOME directory is the user's home directory:

- Windows C:\{user}\.kettle
- or for all other Linix based operating systems ($HOME/.kettle).

The directory may change depending on the user who is logged on. Thus, the configuration files that control the behaviour of PDI jobs and transformations are different from user to user. This also applies when running PDI from the Pentaho BI Platform.

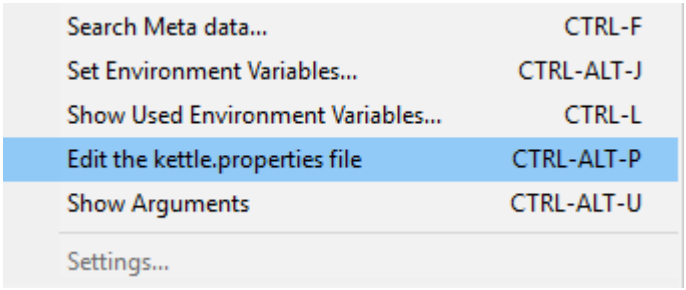
When you set the KETTLE_HOME variable, the PDI jobs and transformations can be run without being affected by the user who is logged on. KETTLE_HOME is used to change the location of the files normally in [user home].kettle.

kettle.properties	Default properties file for variables
shared.xml	Default shared objects file
db.cache	The database cache for metadata
repositories.xml	The local repositories file
.spoonrc	User interface settings, last opened transformation/job
.languageChoice	User language (delete to revert language)

kettle.properties

The *kettle.properties* file is where you will find all the global variables for KETTLE. You can also set global variables that can be used in Transformations and Jobs. For example, you can define database connections, paths to files, or variables that can be used as parameters in your solution.

The kettle.properties can be edited using a Text Editor or via the Toolbar, select:



Each property is denoted as a key/value pair, separated by an = sign.

```
# Note: lines like these with a # in front of it are comments
#
#
#Wed Aug 10 12:29:02 BST 2016
KETTLE_CORE_JOBENTRIES_FILE=
KETTLE_LAZY_REPOSITORY=true
KETTLE_DEFAULT_DATE_FORMAT=
KETTLE_JOB_LOG_SCHEMA=
KETTLE_CHANNEL_LOG_TABLE=
KETTLE_TRANS_PAN_JVM_EXIT_CODE=
KETTLE_SPLIT_FIELDS_REMOVE_ENCLOSURE=false
vfs.sftp.userDirIsRoot=false
KETTLE_COMPATIBILITY_PUR_OLD_NAMING_MODE=N
KETTLE_CARTE_JETTY_ACCEPT_QUEUE_SIZE=
KETTLE MAX LOGGING REGISTRY SIZE=10000
```

Below is an example of connection and path variables that can be added:

connection parameters for db server

DB_HOST=dbhost.domain.org

DR_NAME=database

DB_USER=user

DB_PASSWORD=db_password

path to data files

INPUT_PATH=[path to file]

path to error messages

ERROR_PATH=[path to error file]

These variables can be referenced with the notation: ${key_name} in any configuration field in your Transformations or Jobs.

.languageChoice

- Default language for the PDI client tool

```
#Language Choice
#Mon Nov 21 16:57:12 GMT 2016
LocaleDefault=en_US
LocaleFailover=en_US
```

.spoonrc

Used to store preferences and program state of Spoon. Other Kettle programs do not use this file.

- General settings and defaults
- User interface settings
- The last opened transformation/job

```
ShowBrandingGraphics=N
ShowCanvasGrid=N
ShowCopyOrDistributeWarning=Y
ShowExitWarning=N
ShowHelpToolTips=Y
ShowOSLook54=N
ShowOSLook=N
ShowRepositoriesAtStartup=N
ShowTips=Y
ShowToolTips=Y
ShowWelcomePageOnStartup=N
```

repositories.xml

The information associated with repositories is stored in "repositories.xml". This file is located in the hidden directory ".kettle" in your default home directory. On Windows, the file is located:

C:\Users\{user}\.kettle

shared.xml

A variety of objects can now be placed in a shared objects file on the local machine. The default location for the shared objects file is $HOME/.kettle/shared.xml. Objects that can be shared using this method include:

- Database connections
- Steps
- Slave servers
- Partition schemas
- Cluster schemas

To share one of these objects, simply right-click on the object in the tree control on the left and choose share.

> The location of the shared objects file is configurable on the "Miscellaneous" tab of the Transformation > Settings dialog.

Guided Demo: KETTLE Configuration

Introduction	Browse several files that influence the behaviour of PDI programs.
Objectives	In this guided demonstration, you will... • Adjust Spoon's memory settings • Setting the Kettle home directory (.kettle) • Kettle home directory can be set to point to shared location

KETTLE Configuration Files

Several files influence the behavior of KETTLE programs; the files generally reside under the home directory of each user in a separate. kettle instance.

1. Browse to the following directory:

   C:\Users\<username>\.kettle

kettle.properties	Default properties file for variables
shared.xml	Default shared objects file
db.cache	The database cache for metadata
repositories.xml	The local repositories file
.spoonrc	User interface settings, last opened transformation/job
.languageChoice	User language (delete to revert language)

Setting KETTLE_HOME

In cases when your migrating, upgrading or setting up projects The location of the .kettle directory can be changed explicitly by setting it in a KETTLE_HOME environmental variable, or in the spoon.bat.

Ensure Pentaho Server service has been stopped.

1. Edit the following file in Wordpad or Notepad++:

   C:\Pentaho\design-tools\data-integration\spoon.bat

2. Locate line 118: The code has been placed on several lines for ease.

```
set OPT=%OPT% %PENTAHO_DI_JAVA_OPTIONS%
"-Dhttps.protocols=TLSv1,TLSv1.1,TLSv1.2"
"-Djava.library.path=%LIBSPATH%"
"-DKETTLE_HOME=%KETTLE_HOME%"
"-DKETTLE_REPOSITORY=%KETTLE_REPOSITORY%"
"-DKETTLE_USER=%KETTLE_USER%"
```

```
"-DKETTLE_PASSWORD=%KETTLE_PASSWORD%"
"-DKETTLE_PLUGIN_PACKAGES=%KETTLE_PLUGIN_PACKAGES%"
"-DKETTLE_LOG_SIZE_LIMIT=%KETTLE_LOG_SIZE_LIMIT%"
"-DKETTLE_JNDI_ROOT=%KETTLE_JNDI_ROOT%"
"-Dpentaho.installed.licenses.file=%PENTAHO_INSTALLED_LICENSE_PATH%"
```

3. Edit the %KETTLE_HOME% variable: "-DKETTLE_HOME= [path to .kettle folder]
4. Restart PDI.

Adding the relevant JDBC drivers

Before you can add a data source to a Pentaho server or client tool, you must copy the appropriate JDBC driver JAR to certain directories. To add support for a database, obtain the correct version of the JDBC driver from your database vendor and copy it to the following locations, depending on which products need to connect to this database:

1. Copy the relevant JDBC drivers to the following directories:
   - Data Integration Server: /pentaho/server/data-integration-server/tomcat/webapps/pentaho/WEB-INF/lib/
   - Data Integration client: /pentaho/design-tools/data-integration/lib

> Ensure that there are no other versions of the same vendor's JDBC driver installed in these directories before copying driver JARs. If there are other versions of the same driver, you may have to remove them to avoid confusion and potential class loading problems. This is of concern when you are installing a driver JAR for a data source that is the same database type as your Pentaho solution Repository.

Install using the JDBC Distribution Tool

1. Download a JDBC driver JAR from your database vendor or a third-party driver developer.
2. Copy the JDBC driver JAR you just downloaded to the pentaho/jdbc-distribution directory.
3. Open a cmd prompt or shell tool, navigate to the pentaho/jdbc-distribution directory and enter one of the following:

   Windows:

   distribute-files.bat <name of JDBC driver JAR>

   Linux:

   ./distribute-files.sh

Set PDI Version Control and Comment Tracking Options

Pentaho Data Integration (PDI) can track versions and comments for jobs, transformations, and connection information when you save them. You can turn version control and comment tracking on or off by modifying their related statements in the repository.spring.properties text file.

1. Exit from the PDI client (also called Spoon).
2. Stop the Pentaho Server.
3. Open the pentaho-server/pentaho-solution/systems/repository.spring.properties file in a text editor.
4. Edit the versioningEnabled and versionCommentsEnabled statements:

   versioningEnabled=true

   versionCommentsEnabled=true

By default, version control and comment tracking are disabled (set to false).

Best Practice: manage your ETL workflows with a 3rd party content management tool, e.g. Github; only uploading the production version into the Repository.

Guided Demo: KETTLE Variables

Introduction	This Guided Demo Introduces the Spoon features and customization options.
Objectives	In this guided demonstration, you will… • Learn about variables. • Set variables in the kettle.properties file.

Variables

Variables can be used throughout Pentaho Data Integration, including in transformation steps and job entries. You define variables by setting them with the **Set Variable** step in a transformation or by setting them in the **kettle.properties** file in the directory.

The way to use them is either by grabbing them using the Get Variable step or by specifying meta-data strings like:

${VARIABLE}

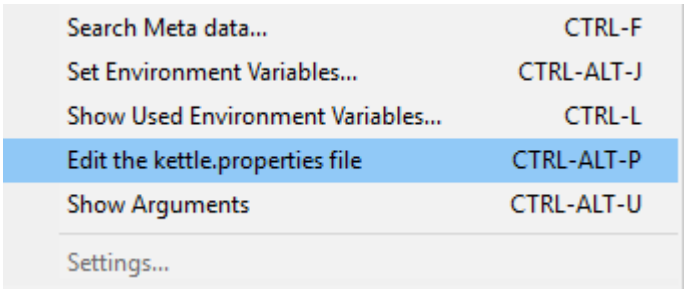
or:

%%VARIABLE%%

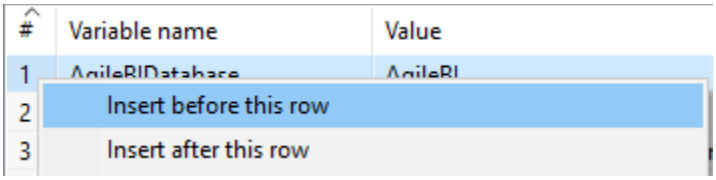
Both formats can be used and even mixed, the first is a UNIX derivative, the second is derived from Microsoft Windows. Dialogs that support variable usage throughout Pentaho Data Integration are visually indicated using a blue dollar sign. You can use <CTRL>+ space hot key to select a variable to be inserted into the property value. Mouse over the variable icon to display the shortcut help.

To set kettle variables for the training directories:

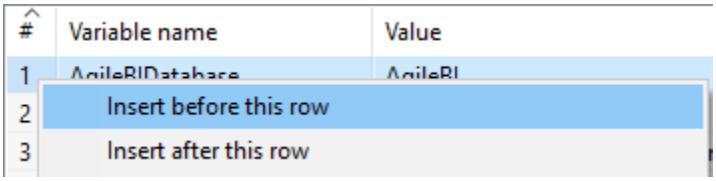
1. Select Edit > Edit the kettle.properties file

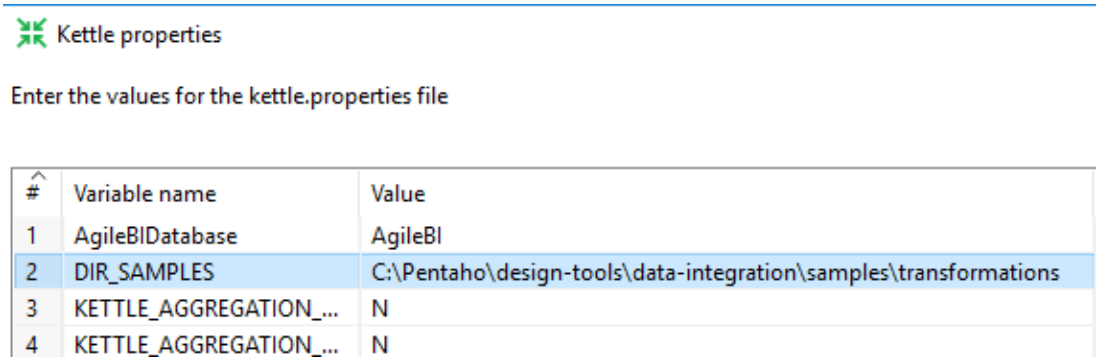


2. Highlight the first row and right mouse click, and select the following option.



3. Insert the following variables:





4. Click OK to save changes.
5. Restart PDI.

This global variable can be used in any Transformation / Job. Remember, as its set in the kettle.properties file, this will have to be copied or recreated if you change server.

- The global variable is accessed by selecting Ctrl + Spacebar, wherever you see a blue diamond

Pentaho Data Integration Repository

There are 3 repository options available:

- File Repository - stored as xml on a server
- Database Repository - can run a script MS SQL Server, Oracle, MySql,
- Enterprise Repository - runs on Data Integration Server, open source CMS Apache Jackrabbit

Pentaho Enterprise Repository:

The Enterprise repository, includes several resources:

- CMS – Apache Jackrabbit implementing the JCR standards (Content Repository API for Java).
- Apache Tomcat with scheduler
- Enterprise security:
    - Configurable authentication including support for LDAP and MSAD
    - Task permissions defining what actions a user/role can perform such as read/execute content, create content, and administer security
    - Granular permissions on individual files / folders
    - Full revision history
    - Ability to lock transformations/jobs
- Web Services – Carte replacement

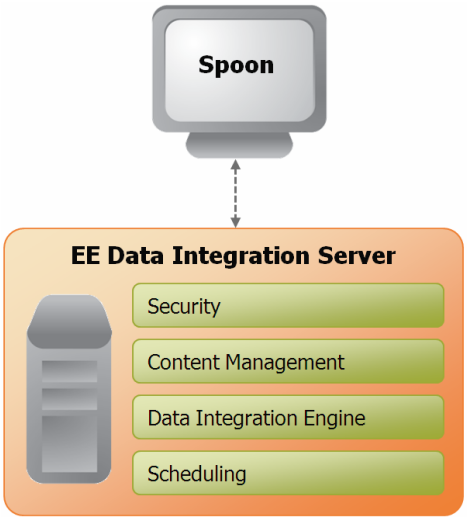
Database Repository:

The KETTLE database repository serializes the metadata to a relational database schema. For example, there is a table called R_TRANSFORMATION which contains the name. description, and extended description of a Transformation.

The script can run on MS SQL Server, Oracle and MySql.

File Repository:

The KETTLE file repository type simply uses stores the XML. It uses Apache VFS driver, to access the metadata, which means you are not limited to a local disk.



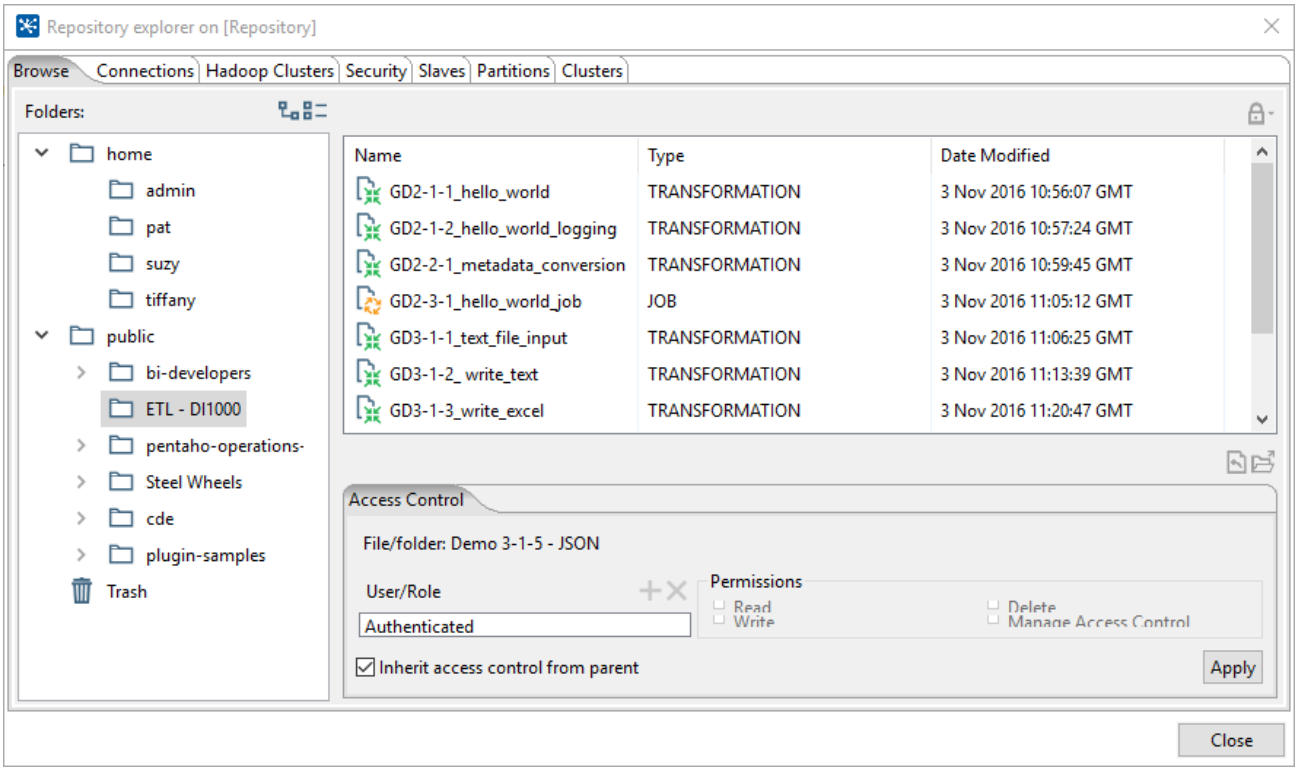
Security – Allows you to manage users and roles (default security) or integrate security into your existing security provider (such as LDAP or Microsoft Active Directory)

Content management – Provides the ability to centrally store and manage ETL jobs and transformations (includes full revision history of content and features such as sharing and locking for collaborative development environments)

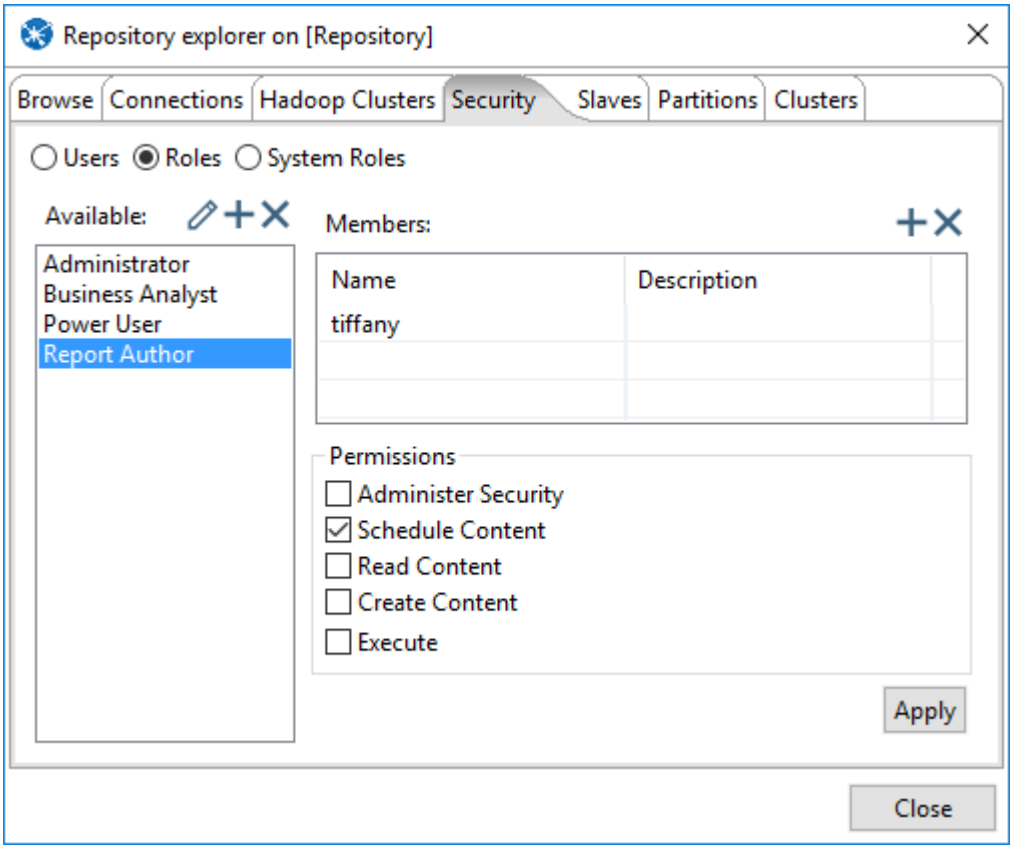
Data Integration Engine – This is a Carte instance. Carte is also used in clustering (covered later in this course).

Scheduling – The Quartz scheduler is used internally, and the tasks are executed in the data integration engine.

You can centrally store ETL objects:



The repository has several predefined roles, with a set of specific rights



Guided Demo: PDI Repository

Introduction	This Guided Demo Introduces the Enterprise Repository features.
Objectives	In this guided demonstration, you will... • Connect to the Enterprise Repository. • If necessary, create an Enterprise Repository.

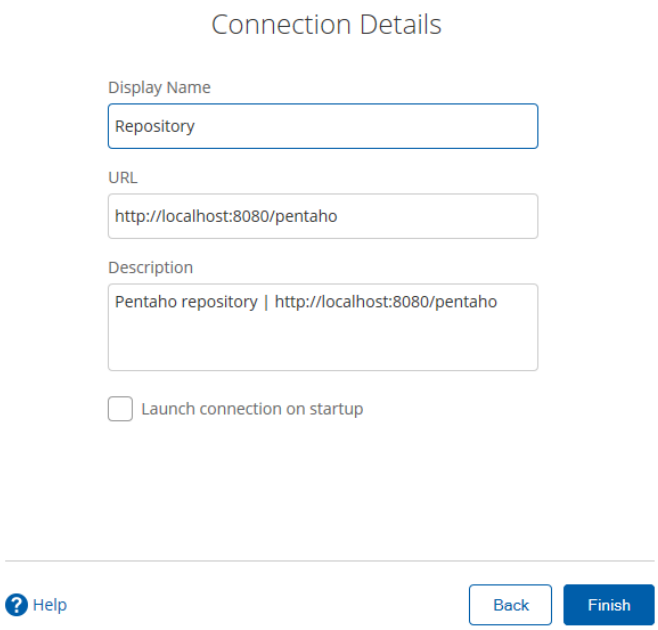
Launching Spoon

When you have large ETL projects, with many ETL developers, then a repository helps coordinate activities.

1. Start Spoon.
2. Click on the Connect button in the top right of the screen.
3. If a connection to the Pentaho Repository has *not already* been created:
   - Click on the **Get Started** button. This option will configure an Enterprise Repository.

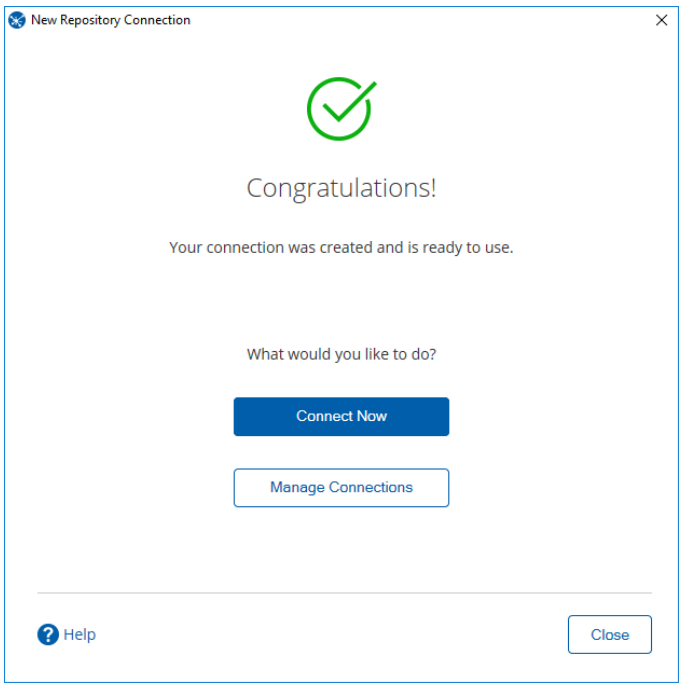
If you wish to configure either Database or File Repository, Click on the 'Other Repositories' link.

4. In the Repository Configuration window, enter a Display Name for your repository.



- Modify the URL associated with your repository if necessary

5. Click Finnish to exit the Repository Configuration dialog box.

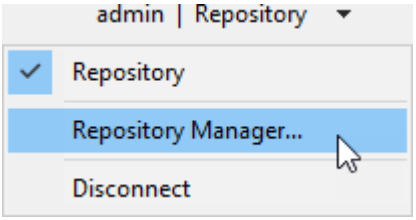


6. Click on Connect Now.
7. Log on to the repository by entering the following credentials:
   - user name = **admin**
   - password = **password**

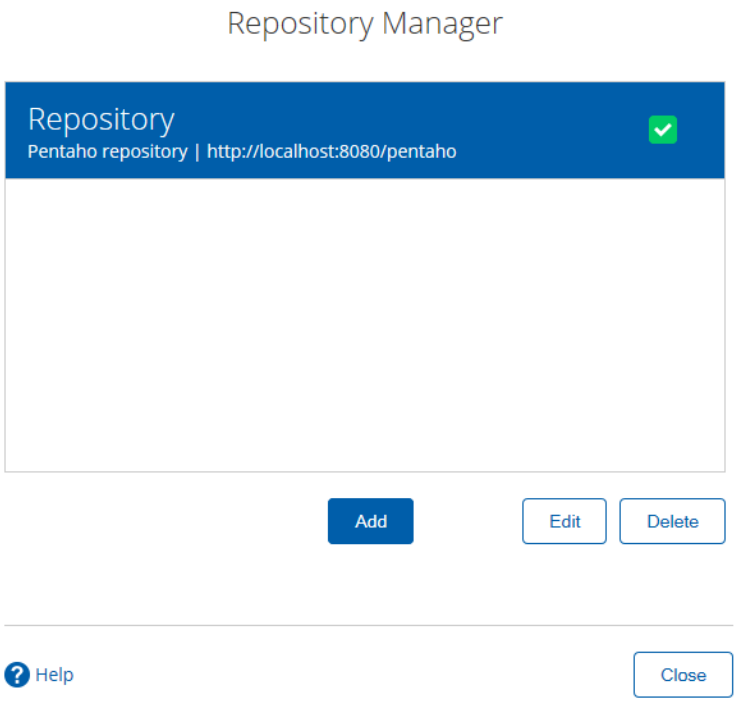
Repository Manager

If you wish to manage your Repository Connections:

1. From the drop-down menu, select repository Manager



2. Select / Add / Edit / Delete the connection by highlighting your Repository Connection and selecting Edit.



Green tick mark indicates 'Launch Connection on Startup'.

Guided Demo: Import / Export PDI Content (optional)

Introduction	You can import and export PDI content to and from a repository by using PDI's built-in functions, explained in these subsections.
Objectives	In this guided demonstration, you will... • Import Content to the Enterprise Repository. • If necessary, create an Enterprise Repository.

Import Content to the Repository

Follow the instructions below to import the repository. You must already be logged into the repository in Spoon before you perform this task.

1. In Spoon, go to Tools > Repository > Import Repository.
2. Locate the export (XML) file that contains the solution repository contents (Repository.xml).
3. Click Open. The Directory Selection dialog box appears.
4. Select the directory in which you want to import the repository.
5. Click OK.
6. Enter a comment, if applicable.
7. Wait for the import process to complete.
8. Click Close.

The full contents of the repository are now in the directory you specified.

Export Content from the Repository

Follow the instructions below to export the repository. You must already be logged into the repository through Spoon to complete this task.

1. In Spoon, go to Tools > Repository > Export Repository.
2. In the Save As dialog box, browse to the location where you want to save the export file.
3. Type a name for your export file in the File Name text box.
4. Note: The export file will be saved in XML format regardless of the file extension used.
5. Click Save.

The export file is created in the location you specified. This XML file is a concatenation of all the data integration content you selected. It is possible to break it up into individual KTR and KJB files by hand or through a transformation.

> Among other purposes, these procedures are useful for backing up and restoring content in the solution repository. However, users, roles, permissions, and schedules will not be included in import/export operations.

Summary

In this first module, you were introduced to Pentaho Data Integration. Specifically, you learned about:

- Pentaho Data Integration platform
- Navigating Spoon Interface
- KETTLE Configuration Files
- Adding JDBC Driver
- Repository

The Pentaho Data Integration platform enables organizations to integrate, blend, convert and transform data from any data source across their entire enterprise. The platform provides the extract, transform and load functionality necessary to integrate a wide variety of data sources, including relational databases, enterprise applications, files and big data. The platforms' main components are:

- Carte
- Pan & Kitchen
- Spoon
- Pentaho Server
- Repository

Carte - is a simple web server that allows you to execute transformations and jobs remotely.

Spoon - is a graphical user interface that allows you to design transformations and jobs that can be run with the Kettle tools — Pan and Kitchen.

Pan - is a data transformation engine that performs a multitude of functions such as reading, manipulating, and writing data to and from various data sources.

Kitchen - is a program that executes jobs designed by Spoon in XML or in a database repository. Jobs are usually scheduled in batch mode to be run automatically at regular intervals.

Pentaho Server - comprises the core BA Platform (Analysis, Reporting, Metadata, ETL) packaged in a standard J2EE WAR along with supporting systems for thin-client tools including the Pentaho User Console, Analyzer, Interactive Reporting, Dashboards.

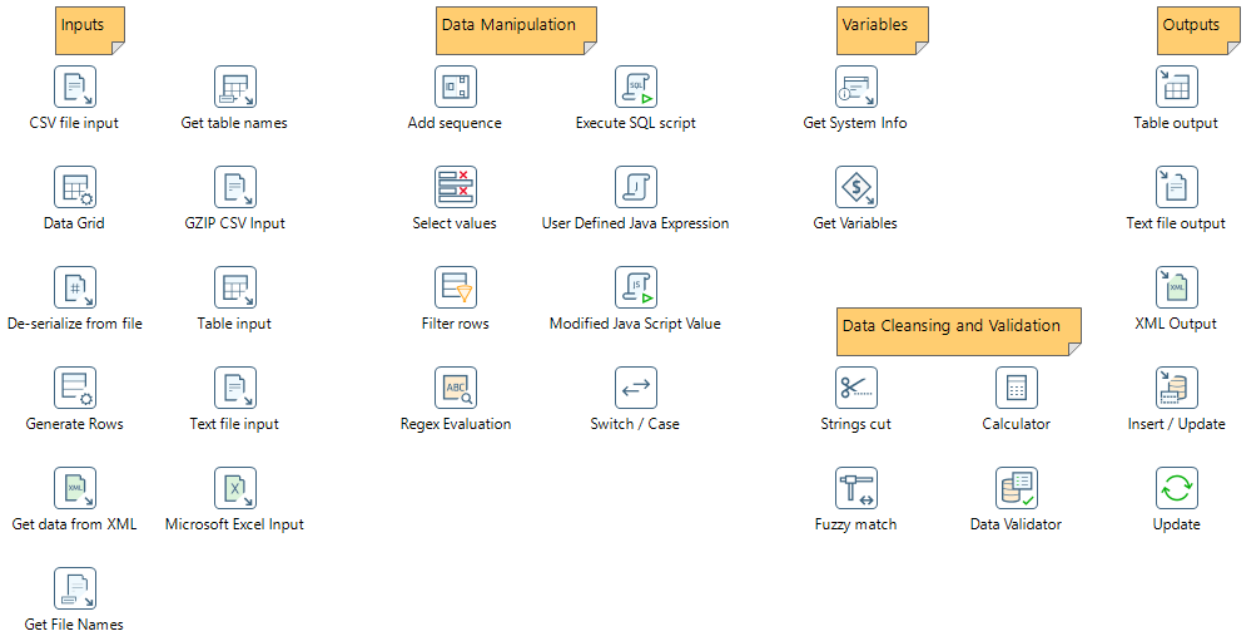
Repository - In addition to storing and managing your jobs and transformations, the Enterprise Repository provides full revision history for Transformations, Jobs, Reports, Dashboards allowing you to track changes, compare revisions and revert to previous versions when necessary.

2: Concepts and Terminology

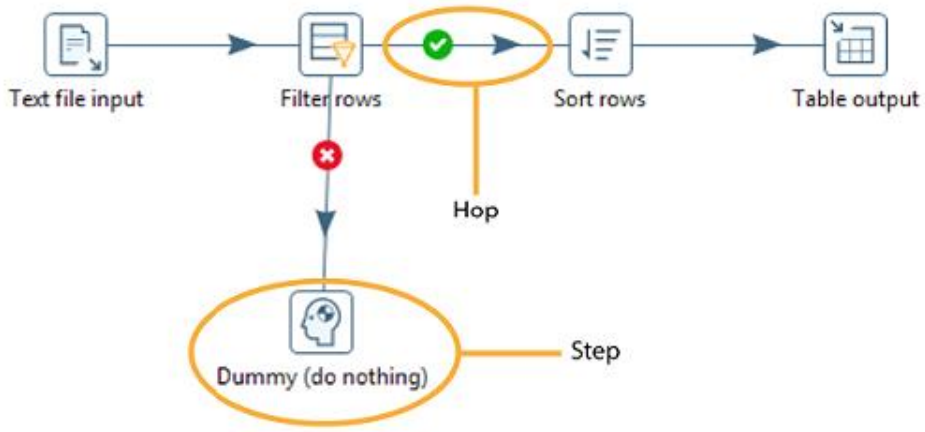
Topics covered in this section:

- Transformations
- Parallelism
- Rows of Data
- Data Types
- Data Inspection
- Jobs

The Data Integration perspective of Spoon allows you to create two basic document types: transformations and jobs. Transformations are used to describe the data flows for ETL such as reading from a source, transforming data and loading it into a target location. Jobs are used to coordinate ETL activities such as defining the flow and dependencies for what order transformations should be run, or prepare for execution by checking conditions such as, "Is my source file available?" or "Does a table exist in my database?"



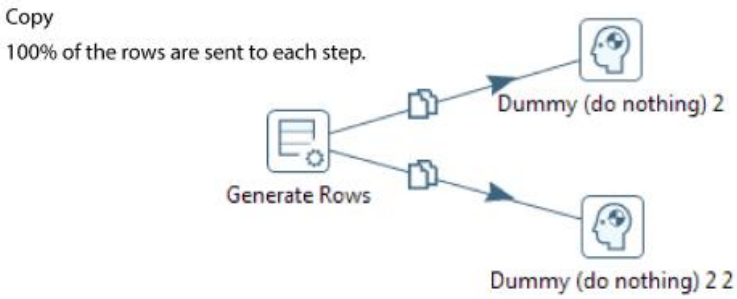
Transformations, Steps & Hops

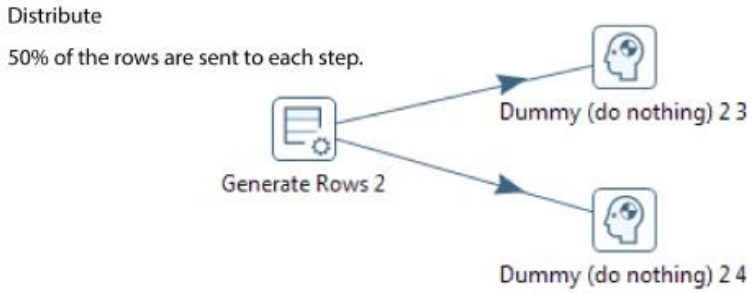


Transformations are the workhorses of the ETL process. They are comprised of Steps, which provide you with a wide range of functionality ranging from reading text-files to implementing slowly changing dimensions. Hops help you define the flow of the data in the stream. They represent a row buffer between the Step Output and the next Step Input, as illustrated in the above Transformation. Data flows from the Text file input step to Filter rows to Sort Rows, finally to Table output.

There are some key characteristics of Steps:

- Step names must be unique in a single Transformation
- Virtually all Steps read and write rows of data (exception Generate rows)
- Most Steps can have multiple outgoing hops. These can be configured to either copy or distribute the data. Copy ensures all Steps receive a copy of the row of data; Distribute sends the data in a round robin fashion to each of the Steps.





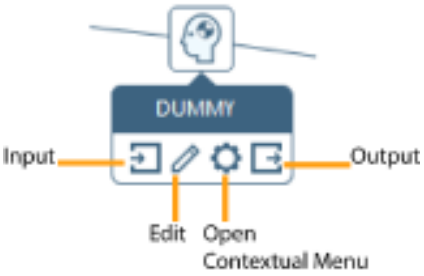
- Steps run in their own thread. It's possible to run multiple copies of the Step, for performance tuning, each in their own thread.
- All Steps are executed in parallel, so it's not possible to define an order of execution.

In addition to Steps and Hops, Notes enable you to document the Transformation.

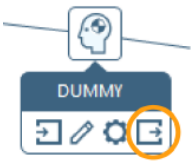
List of Steps:

http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+Steps

You can connect steps together, edit steps, and open the step contextual menu by clicking to edit a step. Click the down arrow to open the contextual menu.



A hop connects one transformation step or job entry with another. The direction of the data flow is indicated by an arrow. To create the hop, click the source step, then press the <SHIFT> key down and draw a line to the target step. Alternatively, you can draw hops by hovering over a step until the hover menu appears. Drag the hop painter icon from the source step to your target step.



Additional methods for creating hops include:

- Click on the source step, hold down the middle mouse button, and drag the hop to the target step.
- Select two steps, then choose New Hop from the right-click menu.
- Use <CTRL + left-click> to select two steps the right-click on the step and choose New Hop.

To split a hop, insert a new step into the hop between two steps by dragging the step over a hop. Confirm that you want to split the hop. This feature works with steps that have not yet been connected to another step only.

Loops are not allowed in transformations because Spoon depends heavily on the previous steps to determine the field values that are passed from one step to another. Allowing loops in transformations may result in endless loops and other problems. Loops are allowed in jobs because Spoon executes job entries sequentially; however, make sure you do not create endless loops.

Mixing rows that have a different layout is not allowed in a transformation; for example, if you have two table input steps that use a varying number of fields. Mixing row layouts causes steps to fail because fields cannot be found where expected or the data type changes unexpectedly. The trap detector displays warnings at design time if a step is receiving mixed layouts.

PDI transformation files

Despite the .ktr extension, PDI transformations are just XML files. As such, you can explore them inside and recognize different XML elements. Look the following sample text:

```
<?xml version="1.0" encoding="UTF-8"?>
<transformation>
<info>
<name>GD2-1-1_hello_world</name>
<description>Hello World Transformation</description>
<extended_description>
This transformation generates 10 rows with the message Hello World.
</extended_description>
...
</transformation>
```

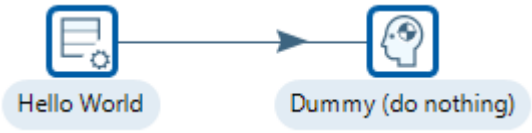
This is an extract from the GD2-1-1_hello_world.ktr file. Here you can see the root element named transformation, and some inner elements such as info and name.

Note that if you copy a step by selecting it in the Spoon canvas and pressing *Ctrl+C*, and then pass it to a text editor, you can see its XML definition. If you copy it back to the canvas, a new identical step will be added to your transformation.

Guided Demo: Hello World

Introduction	In this guided demonstration, we will use Spoon to create a new Transformation that generates some rows of data, with the value "Hello World". The concepts learnt, help to build the foundation necessary for creating any Transformation.

Objectives	In this guided demonstration, you will:
	• Learn to create a new Transformation.
	• Add steps and hops.
	• Configure the following steps:
	  ○ Generate Rows
	  ○ Dummy

Transformation	

To create a new transformation:

1. In Spoon, click File > New > Transformation.

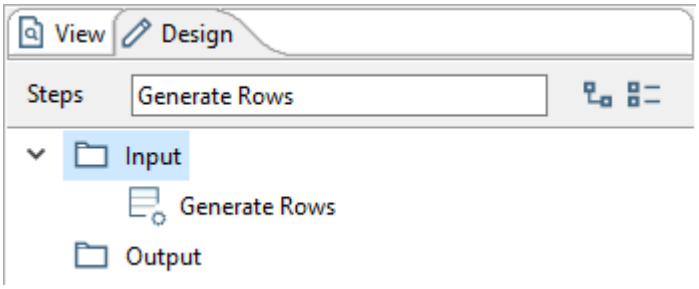
Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

- By clicking New, then Transformation
- By using the CTRL-N hot key

Generate Rows - Hello World

Generate rows outputs a specified number of rows. By default, the rows are empty; however, they can also contain several static fields. This step is used primarily for testing purposes. It may be useful for generating a fixed number of rows, for example, if you require exactly 12 rows for 12 months. Sometimes you may use Generate Rows to generate one row that is an initiating point for your transformation

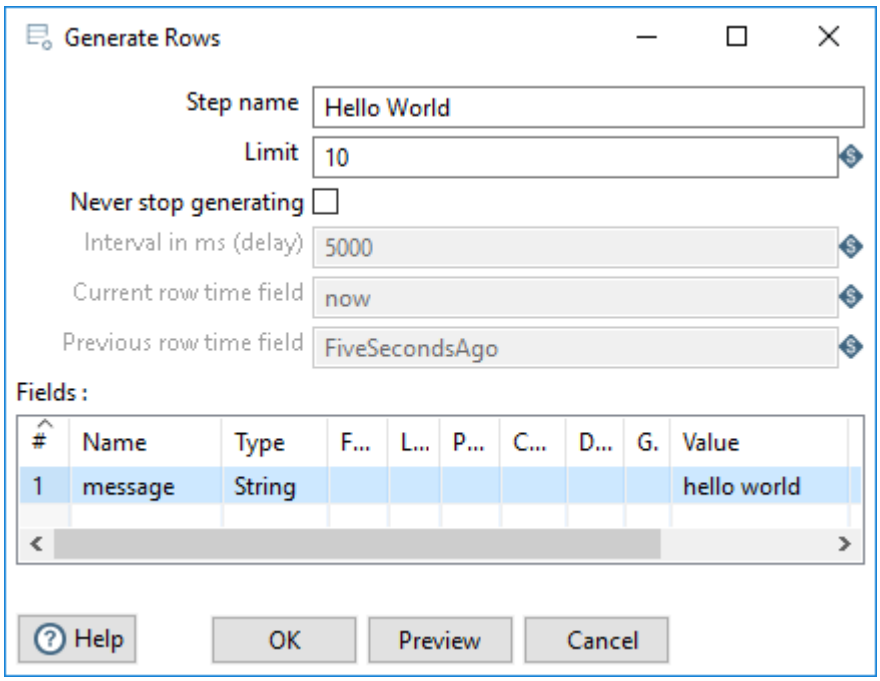
1. To add the Generate Rows step, expand the 'Input' category in the Design tab, and drag the step onto the canvas.
   Alternatively, enter 'Generate Rows' into the search bar.



2. Double-click on the Generate Rows to open step properties.
   Ensure the following details are configured:

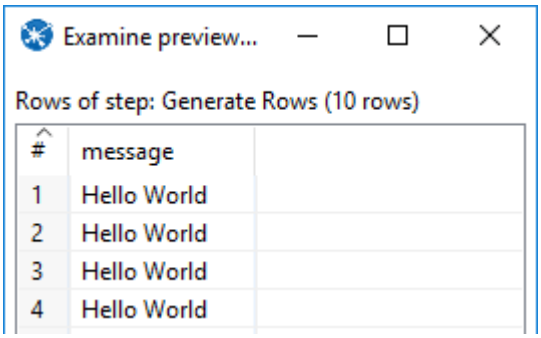
   Step Name: Hello World

   Limit: 10



Before we close this dialog and continue creating the transformation, let's make certain the Step generates the data we expect.

3. Click Preview button. The 'Enter preview size' dialog is displayed.
4. In the 'Enter preview size' dialog, click the [OK] button.



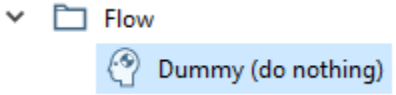
5. Verify 10 rows of data with the message you entered is displayed, and then click the [OK] button to close the 'Examine preview data' dialog.
6. Click OK button to close the 'Generate Rows' dialog.

> Previewing data and testing steps along the way can help to minimize errors and trouble-shooting time later in the transformation creation process.

Dummy

The Dummy step does not do anything. Its primary function is to be a placeholder for testing purposes. For example, to have a transformation, you need at least two steps connected to each other.

1. To add the **Dummy** step, expand the 'Flow' category in the Design tab, and drag the **Dummy** step onto the canvas.



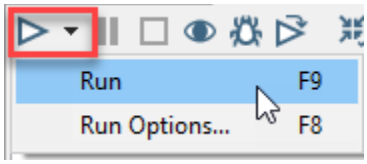
Create a Hop

1. Click on the hello world step.
2. Hold down the Shift key.
3. Drag and drop the hop onto the Dummy step.
4. Release the Shift key.

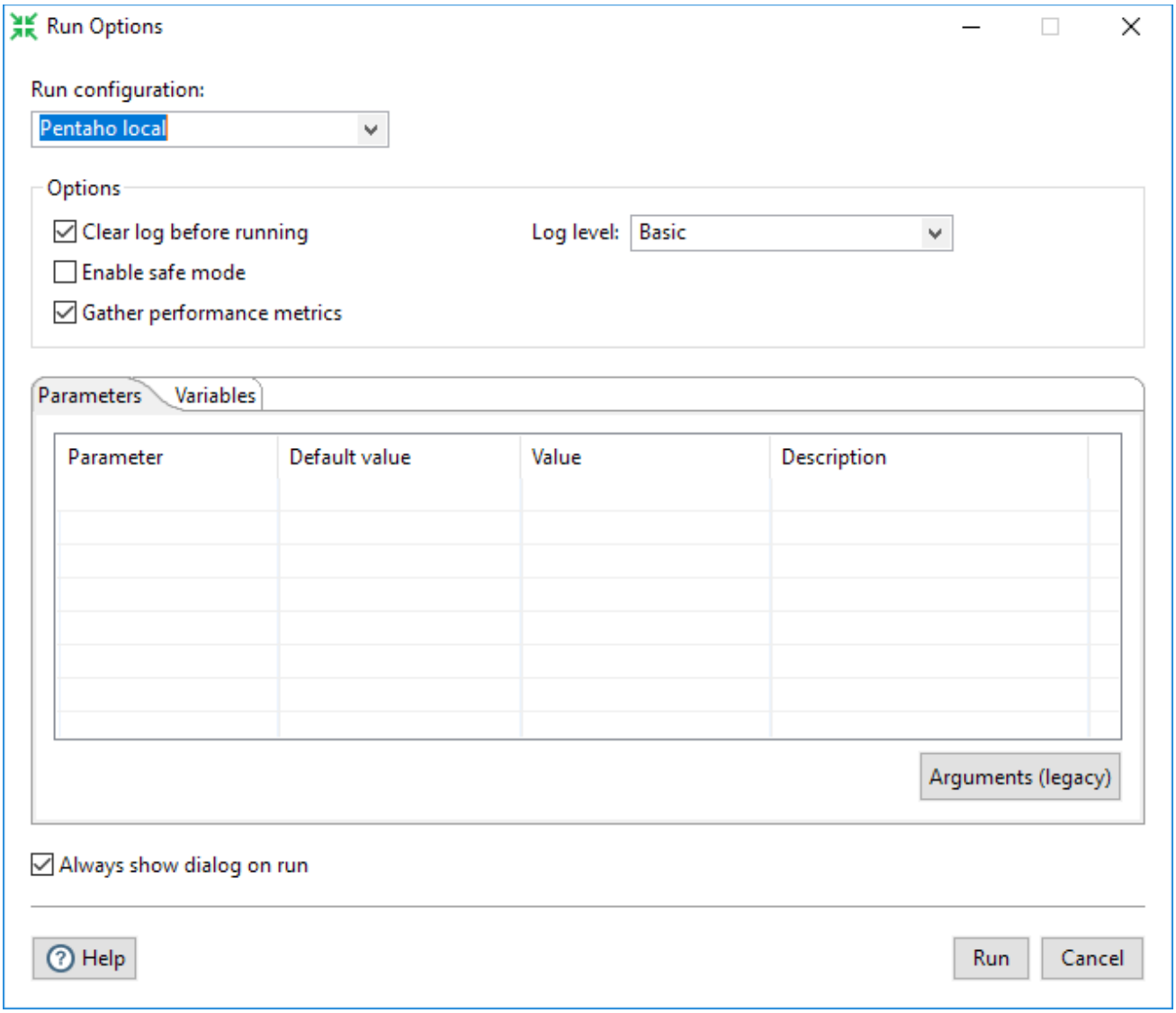
RUN the Transformation

This final part of the creating a transformation focuses exclusively on the local execution option.

1. In Spoon, select Action > Run This Transformation.
   Or Click on the Run button in the toolbar

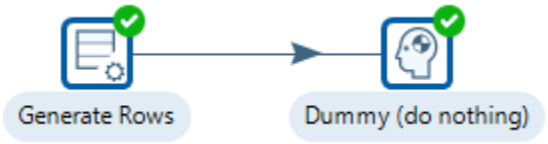


- The Execute a transformation window appears. You can run a transformation locally, remotely, or in a clustered environment. For the purposes of this exercise, keep the default as Local Execution.



2. Click Run.
3. Save the Transformation as: C:Temp\ GD2-1-1_hello_world.ktr

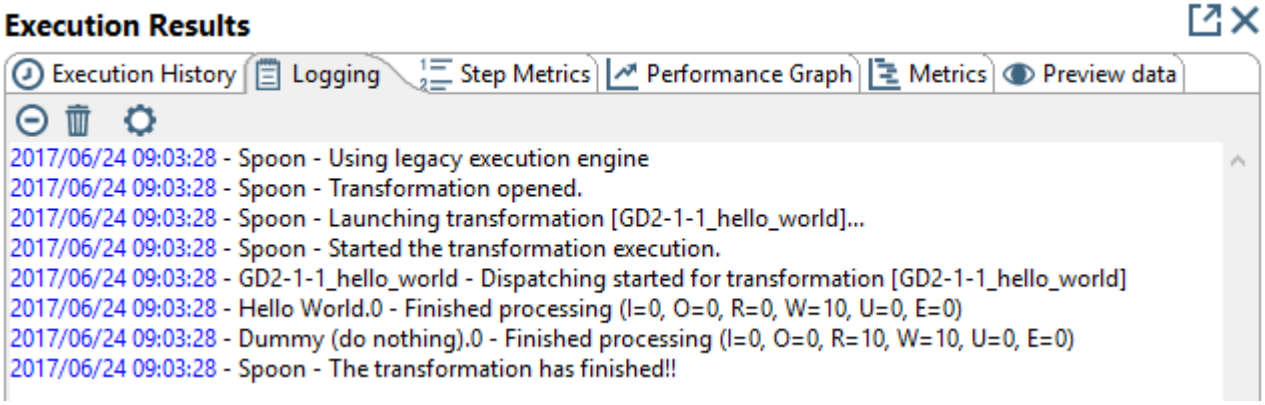
The transformation executes.



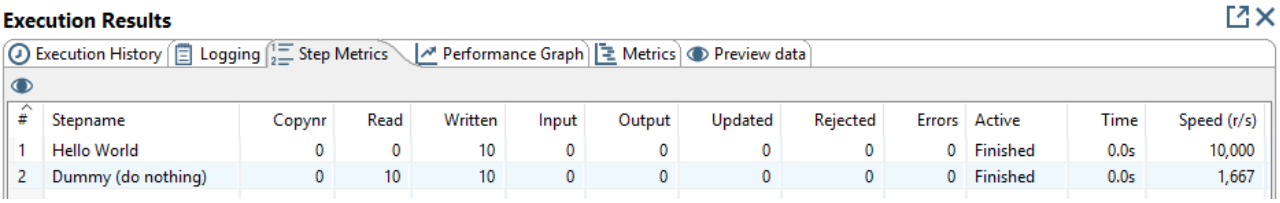
The Execution Results panel opens below the canvas.

The Execution Results section of the window contains several different tabs that help you to see how the transformation executed, pinpoint errors, and monitor performance.

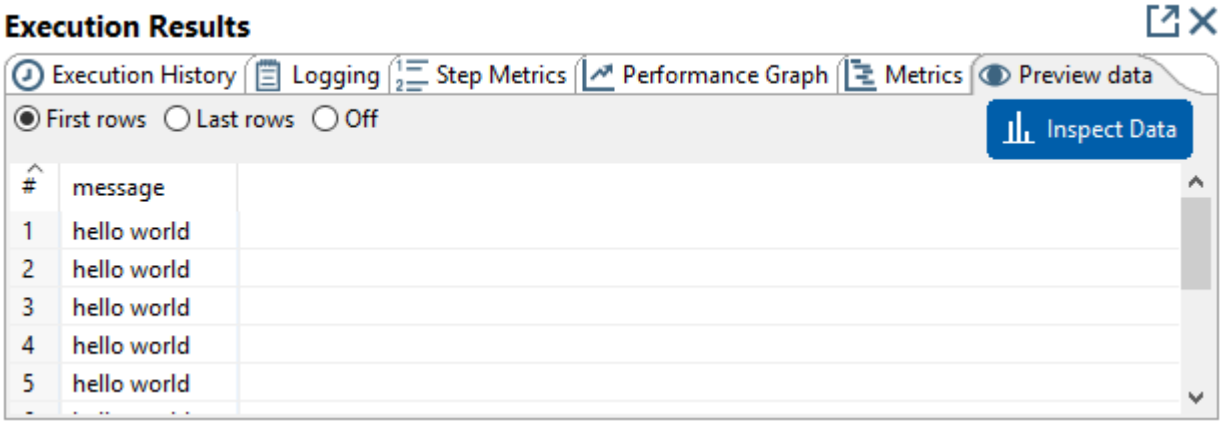
Logging tab displays logging information for each of the steps in the transformation.



Step Metrics tab provides statistics for each step in your transformation including how many records were read, written, caused an error, processing speed (rows per second) and more. This tab also indicates whether an error occurred in a transformation step.



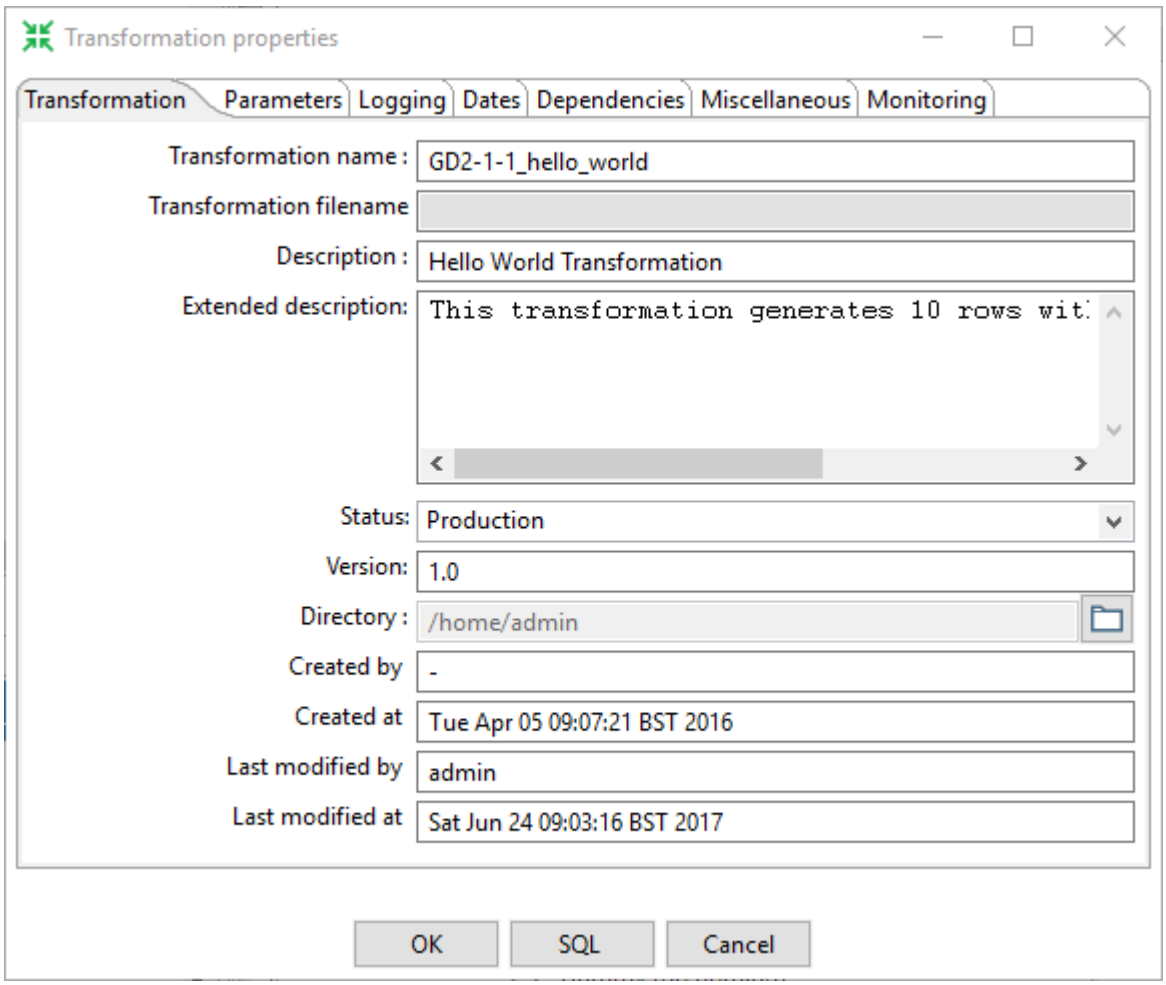
Preview tab displays the records. Click on the Dummy step.



Transformation Properties

To view the transformation properties:

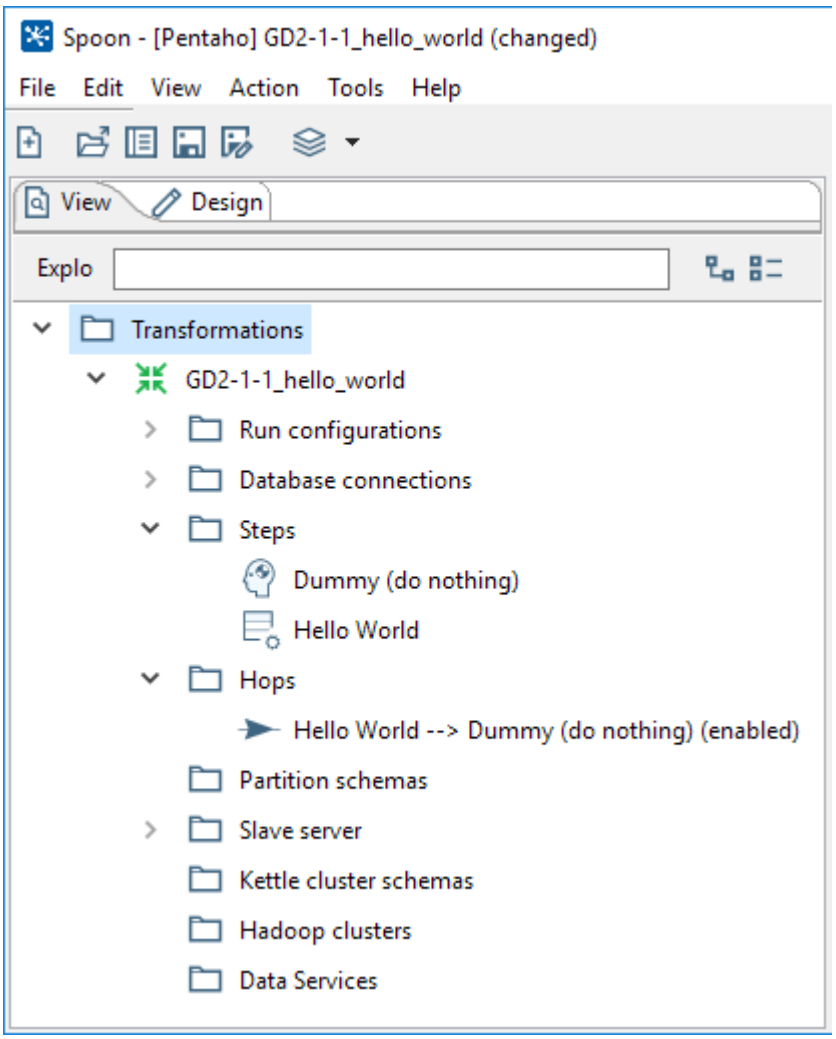
1. Double-click anywhere on the canvas.



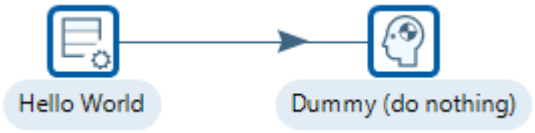
- Optionally, enter a more detailed description in the 'Extended description' property.

Viewing the transformation structure

If you click the View icon in the upper left corner of the screen, the tree will change to show the structure of the transformation currently being edited.

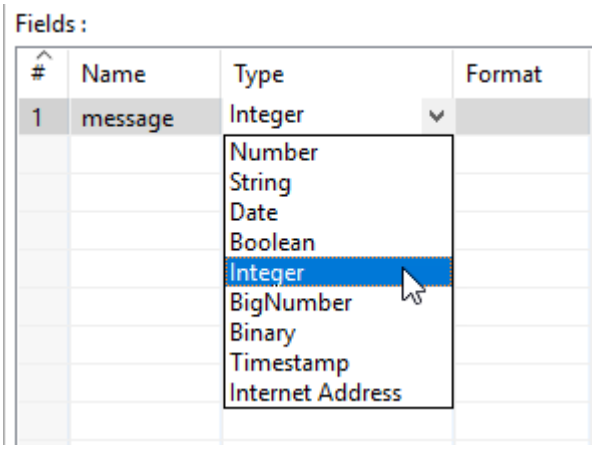


Guided Demo: Hello World (Logging)

Introduction	This guided demonstration introduces basic logging options. The degree of logging can be set from 'Nothing' to 'Row Level'
Objectives	In this guided demonstration, you will: • Change Metadata Data Type • Examine the various levels of logging.
Transformation	

To create an error, change the Type for the field, message, from String to Integer, then view the Execution Results > Logging tab.

1. Double-click on the Generate Rows Step and change the Type, as illustrated below:

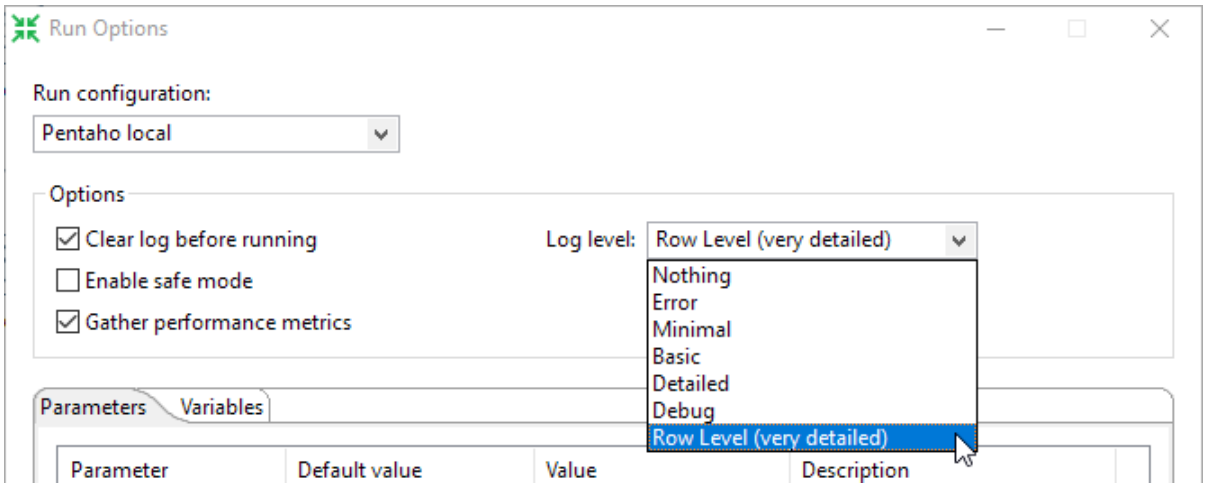


2. Click OK.

Run the Transformation

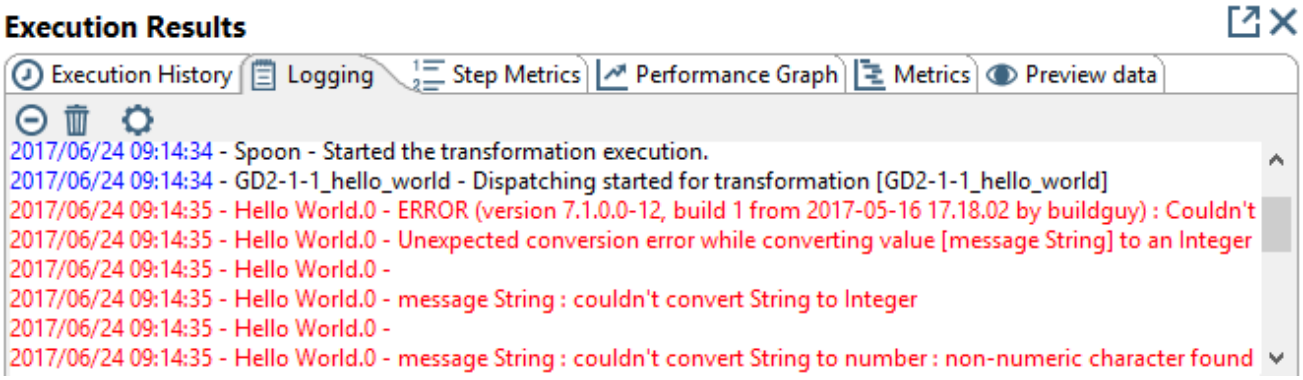
1. Click the RUN button in the Canvas Toolbar.

2. Change the Log level from Basic to Row Level



3. Click Run. The misconfigured Step is highlighted.

4. Click on the Log tab in the Execution Results pane.



The error messages will help pinpoint the issue.

7. Change the Type back to String and re-Run the Transformation.

> Ctrl + C allows you to copy your logs into any text editor.

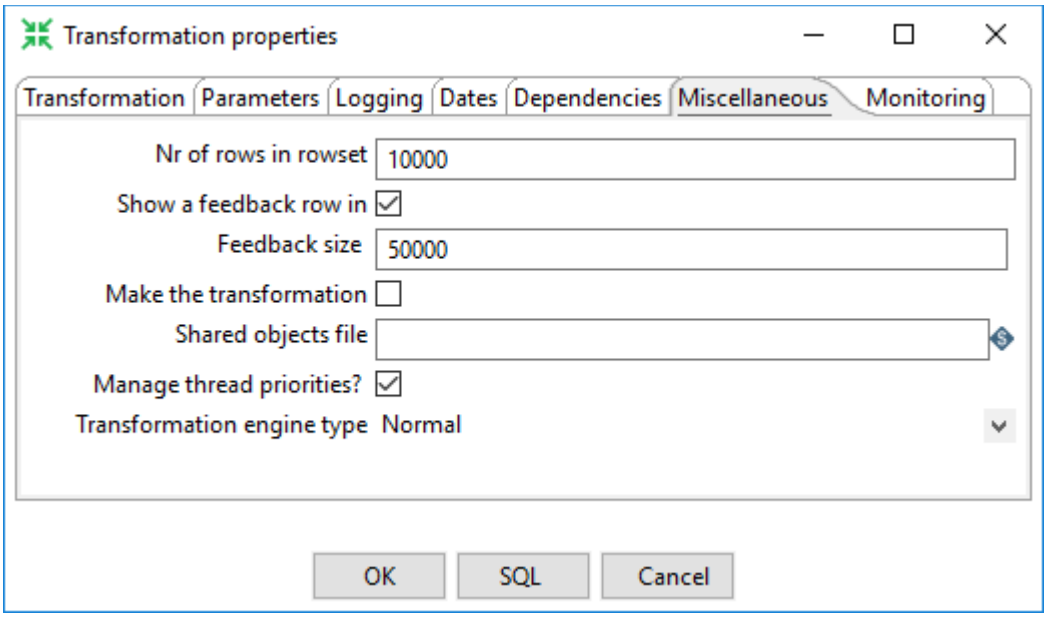
Parallelism

When a transformation starts, all steps start at the same time. The hop is configured as a buffer, with generally a 10k row set.

The flow for the data stream occurs when the first step has initialized, started reading the first row sets, then writing them into the hop (10 k buffer). The row sets are then read by the next step, while the first step is still reading and writing row sets into the stream, and the second step outputs into the stream for the next step, and so on.. The buffer size can be set in Miscellaneous tab, in the Transformation properties panel.

Adjusting the Queue Size

When trying to optimize performance, you may want to adjust the input/output queue size. Especially if you have a lot of RAM available. The queue size is configured as the "Nr of rows in rowset" in the transformation settings and applies to all transformation steps. Increasing it might finish the opening steps of a transformation more quickly, thus freeing up CPU time for the subsequent steps.



Data Types

Every field must have a data type. The data type can be any of the common data types—number (float), string, date, Boolean, integer, or big number. Strings are simple, just text for which you may specify a length.

Data that passes from Step to Step over a Hop flows as rows of data (data stream), in fields. Fields can contain the following data types:

- **String** – any type of character
- **Number** – a double precision floating point number
- **Integer** – signed long integer
- **Big Number** – a number with arbitrary precision
- **Date** – date time with millisecond precision
- **Timestamp** – nanosecond precision
- **Boolean** – true or false value
- **Binary** – contain images, sounds, videos, any binary data
- **Internet** – internet address

Date Fields

Date is one the main data types available in PDI. When a date field is created, you should define the format of the data so that PDI can recognize in the field the different components of the date. There are several formats that may be defined for a date, all of them combinations of letters that represents date or time components. Here are the most basic ones:

Letters	Value
y	Year
M	Month
d	Day
H	Hour (0-23)
m	Minutes
s	Seconds

For a complete reference, check the Sun Java API documentation located at: http://java.sun.com/javase/6/docs/api/java/text/SimpleDateFormat.html.

Numeric Fields

Numeric fields are present in almost all PDI transformations.

There are several formats you may apply to a numeric field. The format is basically a combination of predefined symbols, each with a special meaning. The following are the most used symbols:

Symbol	Value
#	Digit Leading zeros are not shown
0	Digit If the digit is not present, zero is displayed in its place
.	Decimal separator
-	Minus sign
%	Field has to be multiplied by 100 and shown as a percentage

If you don't specify a format for your numbers, you may still provide a Length and

Precision. Length is the total number of significant figures, while precision is the number

of floating-point digits.

If you neither specify format nor length or precision, Kettle behaves as follow. While reading, it does its best to interpret the incoming number, and when writing, it sends the data as it comes without applying any format.

For a complete reference on number formats, you can check the Sun Java API documentation available at http://java.sun.com/javase/6/docs/api/java/text/DecimalFormat.html.

Each Step also able to manipulate the metadata(description) of that row:

- **Name** – name of the field
- **Type** – data type of the field
- **Length** – length of string or number
- **Precision** – decimal precision of number
- **Mask** – representational format (used in data type conversions)
- **Format** – data conversion mask
- **Decimal** – decimal symbol in number
- **Group** – grouping symbol in number
- **Step Origin** – keeps track of the origin of the field in KETTLE

Some considerations to consider when designing your transformation:

- All rows need to have the same layout or structure.
- Beyond data type and name, field metadata is not enforced during the execution of a transformation. This means that a *string* is not automatically cut to the specified length.
- By default, empty *strings* (“”) are considered NULL

Data Conversion

Data Conversion, as the name implies, deals with changes required to convert data from one format to another. This can take place either explicitly, for example in the Select values step, where you can change the data type via the meta-data tab, or implicitly, when you're writing to a database table.

Date to String:

To convert between String and Date data types you only need to specify a conversion mask.

December 6, 2009 at 21 hours, 6 minutes and 54.321 seconds

Conversion Mask	Result
yyyy/MM/dd'T'HH:mm:ss.SSS	2009/12/06T21:06:54.321
H:mm a	9:06 PM
HH:mm:ss	21:06:54
M-d-yy	12-06-09

http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html

Numeric to String:

Conversion of numeric data to and from String uses the following metadata fields.

- Conversion mask
- Decimal symbol
- Grouping symbol
- Currency symbol

The numeric conversion mask determines how the value is represented in a textual format. It has bearing on the actual precision or rounding of the numeric data itself.

Value	Conversion Mask	Decimal Symbol	Grouping Symbol	Result
1234.5678	#,###.##	.	,	1,234.57
1234.5678	0000,000.00000	,	,	001.234,56780
-1.9	#.00;-#.00	.	,	-1.90
1.9	#.00;-#.00	.	,	1.90
12	00000;-00000			00012

http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html

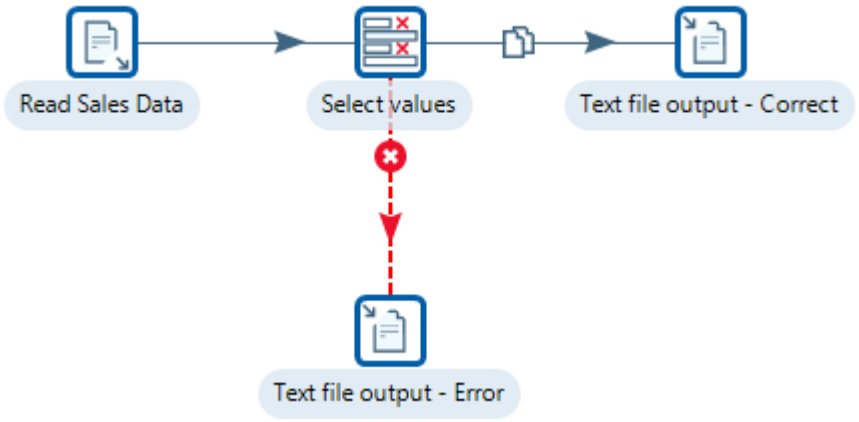
Other Data Type Conversions:

From	To	Description
Boolean	String	Converted to Y or N unless length is 3 or higher: true or false
String	Boolean	Case-insensitive comparison is made: Y, true, yes & 1 is true Any other string is false
Integer **Date**	Date Integer	Number of milliseconds passed since January 1, 1970 00:00:00 GMT Date Date number of milliseconds passed since January 1, 1970 00:00:00 GMT

Guided Demo: Data Conversion & Error Handling

Introduction	To assure this file is ready to load into a Test File Output step, this guided demo introduces the Select values step and error handling to find invalid date values.

Objectives	In this guided demonstration, you will…
	• Explicit data conversion using the Select values step
	• Defining error handling
	• Configure the following steps:
	  ○ CSV File Input
	  ○ Select values
	  ○ Text File output

Transformation	

To create a new transformation:

1. In Spoon, click File > New > Transformation.

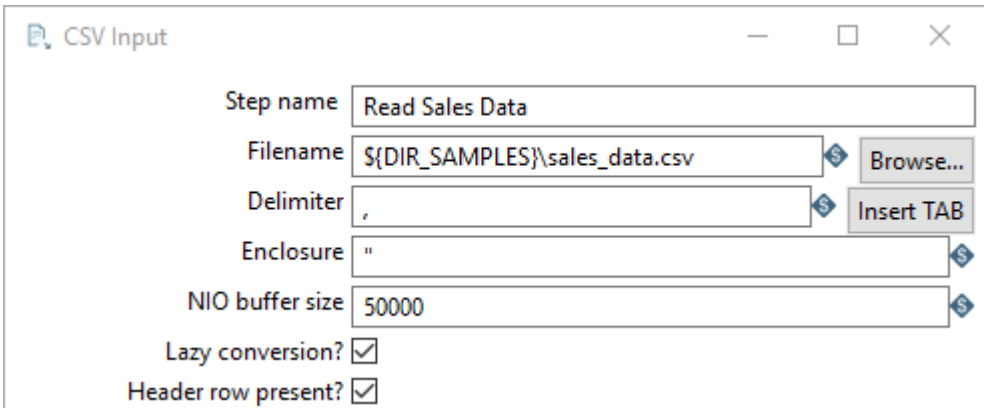
Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

- By clicking New, then Transformation
- By using the CTRL-N hot key

CSV File Input – Read Sales Data

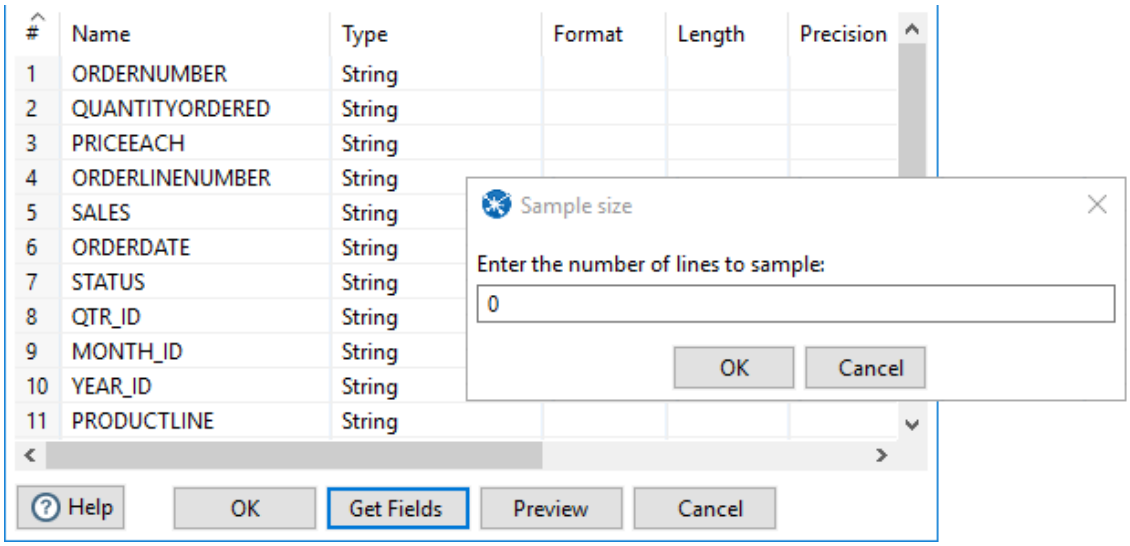
This step provides the ability to read data from a delimited file. The CSV label for this step is a misnomer because you can define whatever separator you want to use, such as pipes, tabs, and semicolons; you are not constrained to using commas. Internal processing allows this step to process data quickly. Options for this step are a subset of the Text File Input step.

1. To add the Read Sales Data step, expand the 'Input' category in the Design tab, and drag the CSV file input step onto the canvas.
2. Open the CSV file input properties dialog box.
3. Rename the Step name to: Read Sales Data
4. Ensure the following details are configured, as outlined below:



- Remember to enclose the global variable in curly brackets.

5. Click on the Get Fields button.



6. In the Sample size dialogue change the size of 100 to 0. This causes the entire file to be read. The scan result dialogue appears. Click the Close button at the bottom of the dialogue.

   The scan results will determine the metadata values based on the algorithm.

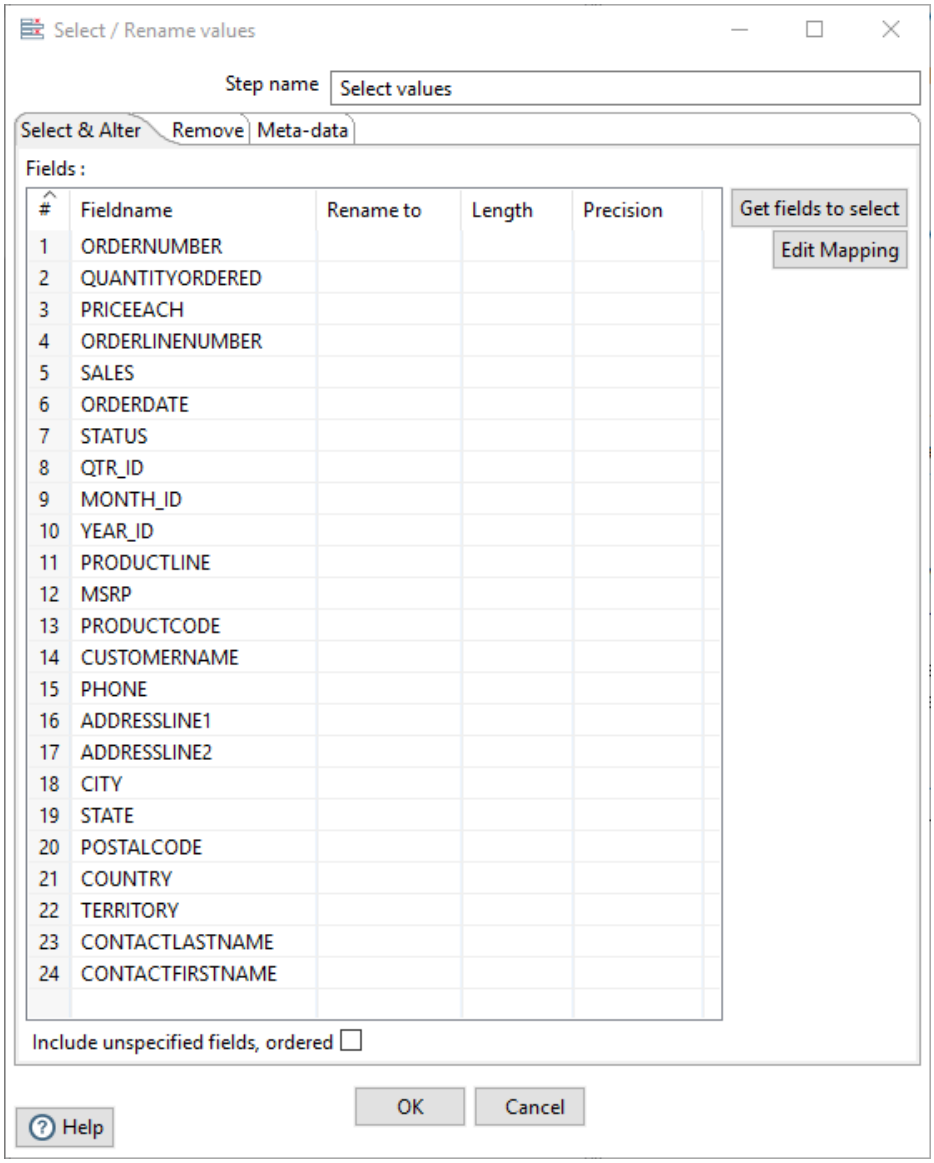
7. Click OK.

Select Values

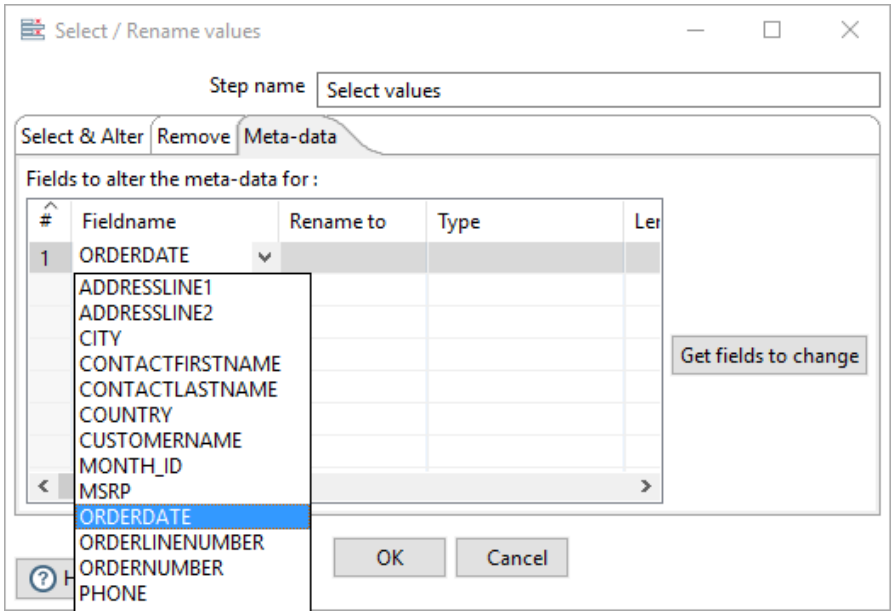
The Select Values step is useful for selecting, removing, renaming, changing data types and configuring the length and precision of the fields on the stream. These operations are organized into different categories:

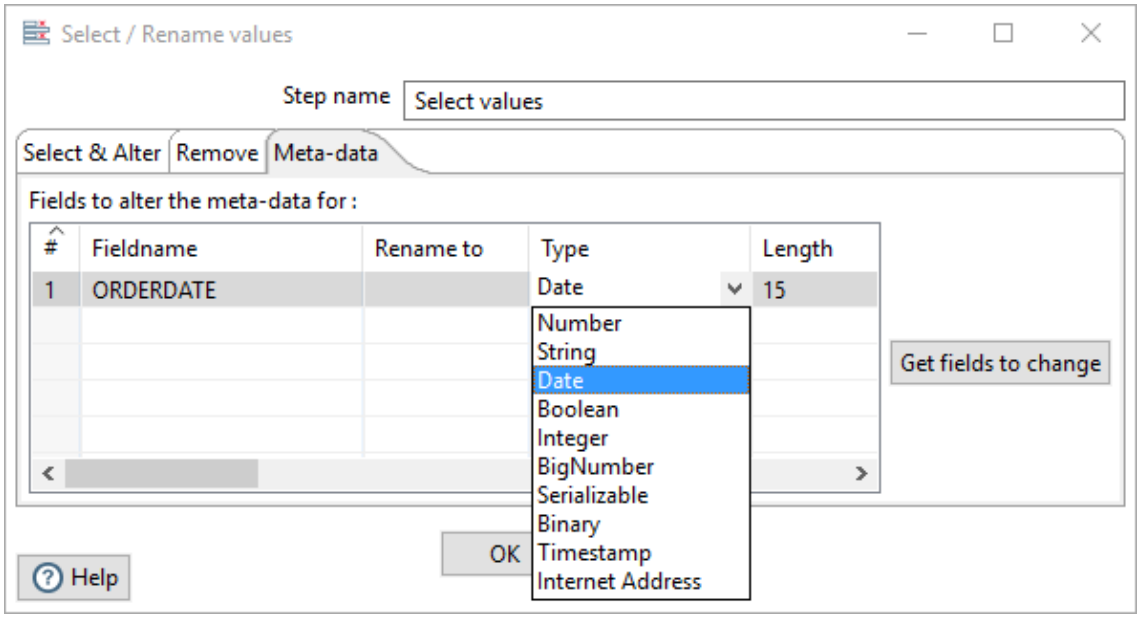
- **Select and Alter** — Specify the exact order and name in which the fields should be placed in the output rows
- **Remove** — Specify the fields that should be removed from the output rows
- **Meta-data** - Change the name, type, length and precision (the metadata) of one or more fields

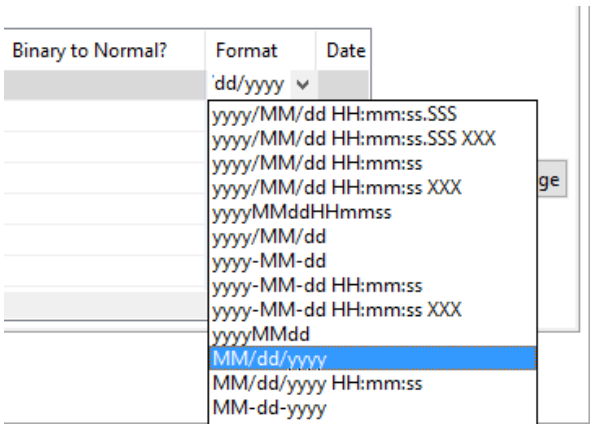
1. To add the Select values step, expand the 'Transform' category in the Design tab, and drag the Select values step onto the canvas.
2. Create a Hop from the Read Sales Data input to the Select values step.
3. To open the step properties dialog, double-click the Select values step.
4. Ensure the tab: Select & Alter is highlighted.
5. Click on the button: Get fields to select



6. Ensure the tab: Meta-data is highlighted. Ensure the following details are configured, as illustrated:







7. Close the step.

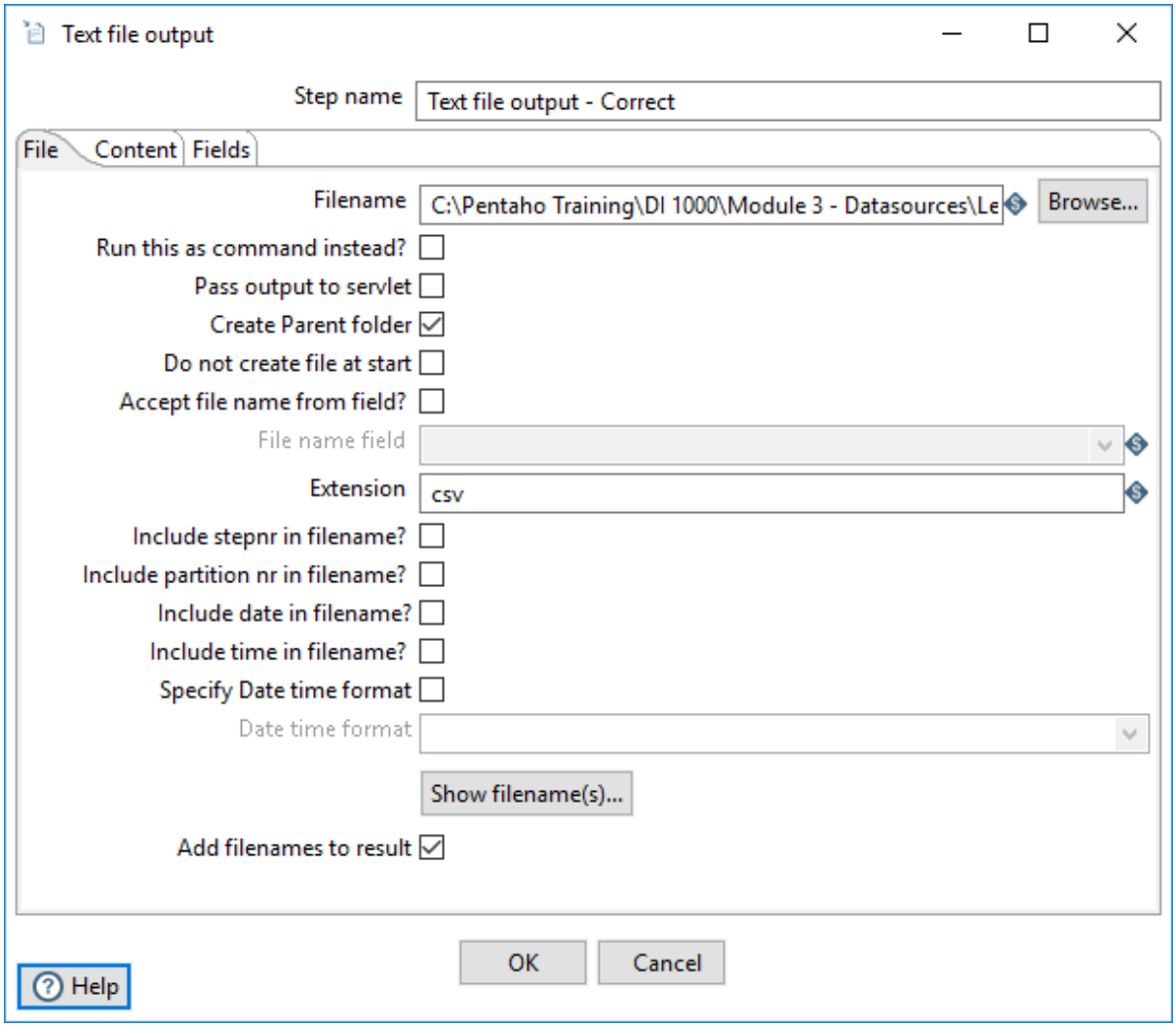
Text File Outputs

The Text file output step is used to export data to text file format. This is commonly used to generate Comma Separated Values (CSV files) that can be read by spreadsheet applications. It is also possible to generate fixed width files by setting lengths on the fields in the fields tab.

> It is not possible to execute this step in parallel to write to the same file. In this case, you need to set the option "Include stepnr in filename" and later merge the files.

Text File Output - Correct

1. To add the Text File Output step, expand the 'Output' category in the Design tab, and drag the Text File Output step onto the canvas.
2. Create a Hop from the Select values to the Text File Output step.
3. To open the step properties dialog, double-click the Text File Output step.
4. Ensure the following details are configured:



5. Click on the Fields tab.
6. Click on the Button: Get fields.
7. Close step.

Text File Output - Error

1. To add the Text File Output step, expand the 'Output' category in the Design tab, and drag the Text File Output step onto the canvas.
2. Create a Hop from the Select values to the Text File Output step.
3. To open the step properties dialog, double-click the Text File Output step.
4. Ensure the following details are configured:

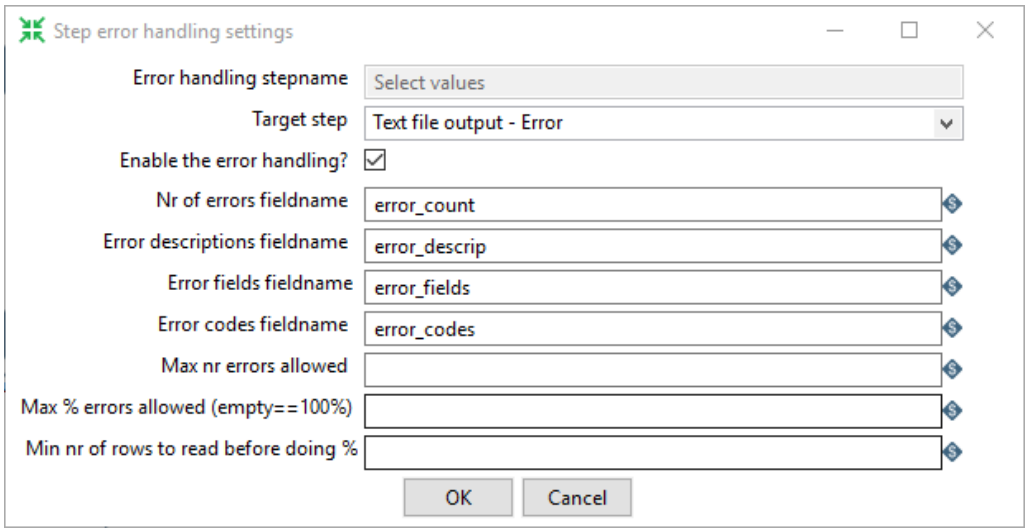
File tab:

Stepname	Error Text file output
Filename	[path to demo folder]\sales_data_error
Extension	csv
Add filenames to result	checked

5. Click on the Fields tab.
6. Click on the Button: Get fields.
7. Close step.

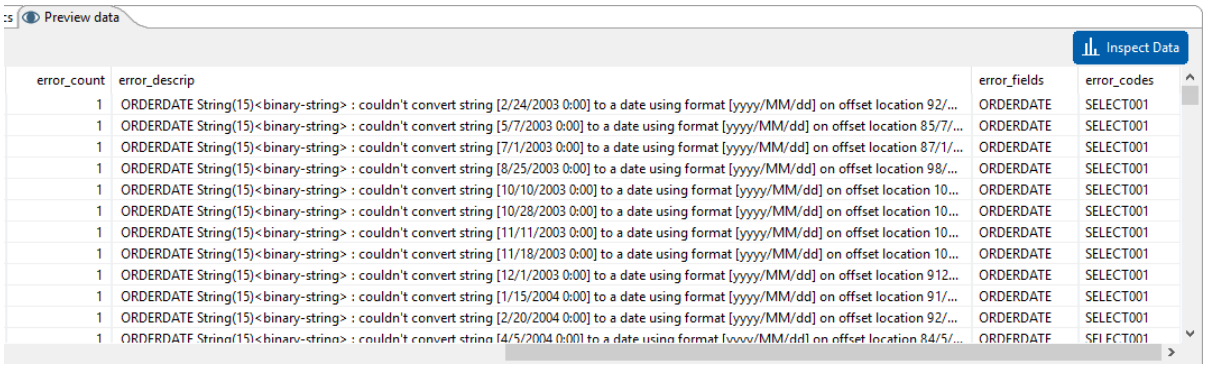
Configure the Hop for Error Handling

1. Double-click on the Error hop, and configure as illustrated below:



Run the Transformation

1. Preview the Correct step.
2. Change the date format to: yyyy-MM-dd
3. Scroll to the end, in the Preview data tab.



error_count	error_descrip	error_fields	error_codes
1	ORDERDATE String(15)<binary-string> : couldn't convert string [2/24/2003 0:00] to a date using format [yyyy/MM/dd] on offset location 92/...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [5/7/2003 0:00] to a date using format [yyyy/MM/dd] on offset location 85/7/...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [7/1/2003 0:00] to a date using format [yyyy/MM/dd] on offset location 87/1/...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [8/25/2003 0:00] to a date using format [yyyy/MM/dd] on offset location 98/...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [10/10/2003 0:00] to a date using format [yyyy/MM/dd] on offset location 10...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [10/28/2003 0:00] to a date using format [yyyy/MM/dd] on offset location 10...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [11/11/2003 0:00] to a date using format [yyyy/MM/dd] on offset location 10...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [11/18/2003 0:00] to a date using format [yyyy/MM/dd] on offset location 10...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [12/1/2003 0:00] to a date using format [yyyy/MM/dd] on offset location 912...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [1/15/2004 0:00] to a date using format [yyyy/MM/dd] on offset location 91/...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [2/20/2004 0:00] to a date using format [yyyy/MM/dd] on offset location 92/...	ORDERDATE	SELECT001
1	ORDERDATE String(15)<binary-string> : couldn't convert string [4/5/2004 0:00] to a date using format [yyyy/MM/dd] on offset location 84/5/...	ORDERDATE	SELECT001

4. Save Transformation: GD2-2-1_metadata_conversion.ktr

As the data stream has been 'split' you will need to determine how the rows are distributed.

In Pentaho, the data movement can be controlled in three ways:

Copy Data – all rows are sent to the destination step

Distribute Data – destination steps receive rows in turn

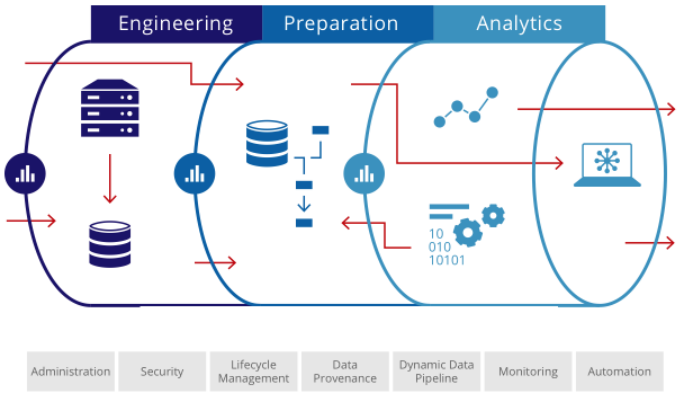
Custom Row Distribution – for example load balancing

The options are available in PDI by *right clicking* on any of the Kettle Steps and selecting the *Data Movement* Option.

Data Explorer

Imagine being able to access analytics from anywhere in the data pipeline, not just after data has been prepared and exported to a business analytics tool. Leverage the only platform on the market that brings analytics into data prep, without the need to switch in and out of tools– so that you can shorten the cycle from data to insights.

- Bringing Analytics into Data Prep: ETL developers and data prep staff can spot check analytics in-flight with access to charts, graphs, visualizations, or ad hoc analysis from any step in the data prep process.
- Share Analytics During Data Prep: Publish data sources for the business while preparing data. With the ability to immediately share data sources, IT can better collaborate with the business for a quicker, less iterative approach to the right analytics.



When working with your transformation, you can gain valuable insights into the data of most steps through visualizing and interacting with your data in many ways. The ability to quickly inspect step data reduces the amount of iterative work needed while building your transformation and enables you to rapidly publish a data source to share with either your teams or across your organization.

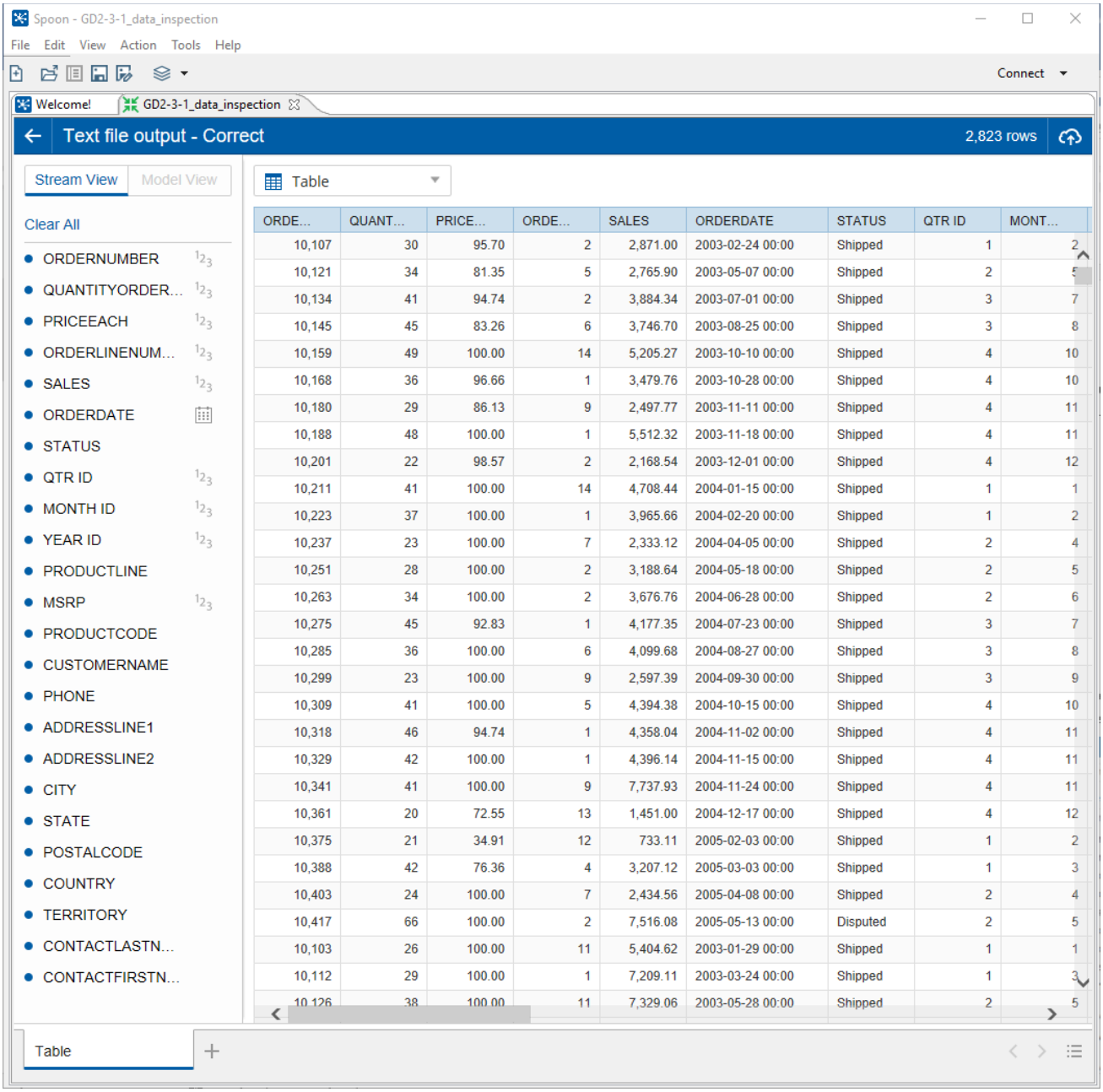
Begin inspecting your transformation by clicking on a step. This displays the **fly-out inspection bar** at the top of the canvas area. The bar displays the name of the step selected and offers two options:

- **Inspect Data** - Lets you inspect the data of a step once the transformation has run.
- **Run and Inspect Data** - Runs the transformation, then lets you inspect the data of a step. Additionally, you can begin inspecting in the following ways:
- **Step Context Menu** - Right-click on a step and choose either Inspect Data or Run and Inspect Data.
- **Preview Data Panel** - Select the Preview Data tab. Click the Inspect Data button located at the top right of the Preview Data bar.
- **Actions Menu** - Select a step. From the Menu bar, click Action>Inspect Data or Action > Run and Inspect Data.
- **Keyboard Shortcuts -** Select a step. Then using your keyboard:
    - In Windows, press either **Shift+Ctrl+F9** (Inspect Data) or **Ctrl+F9** (Run and Inspect Data).
    - In OS X, press **Shift+Command+F9** (Inspect Data) or **Command+F9** (Run and Inspect Data).

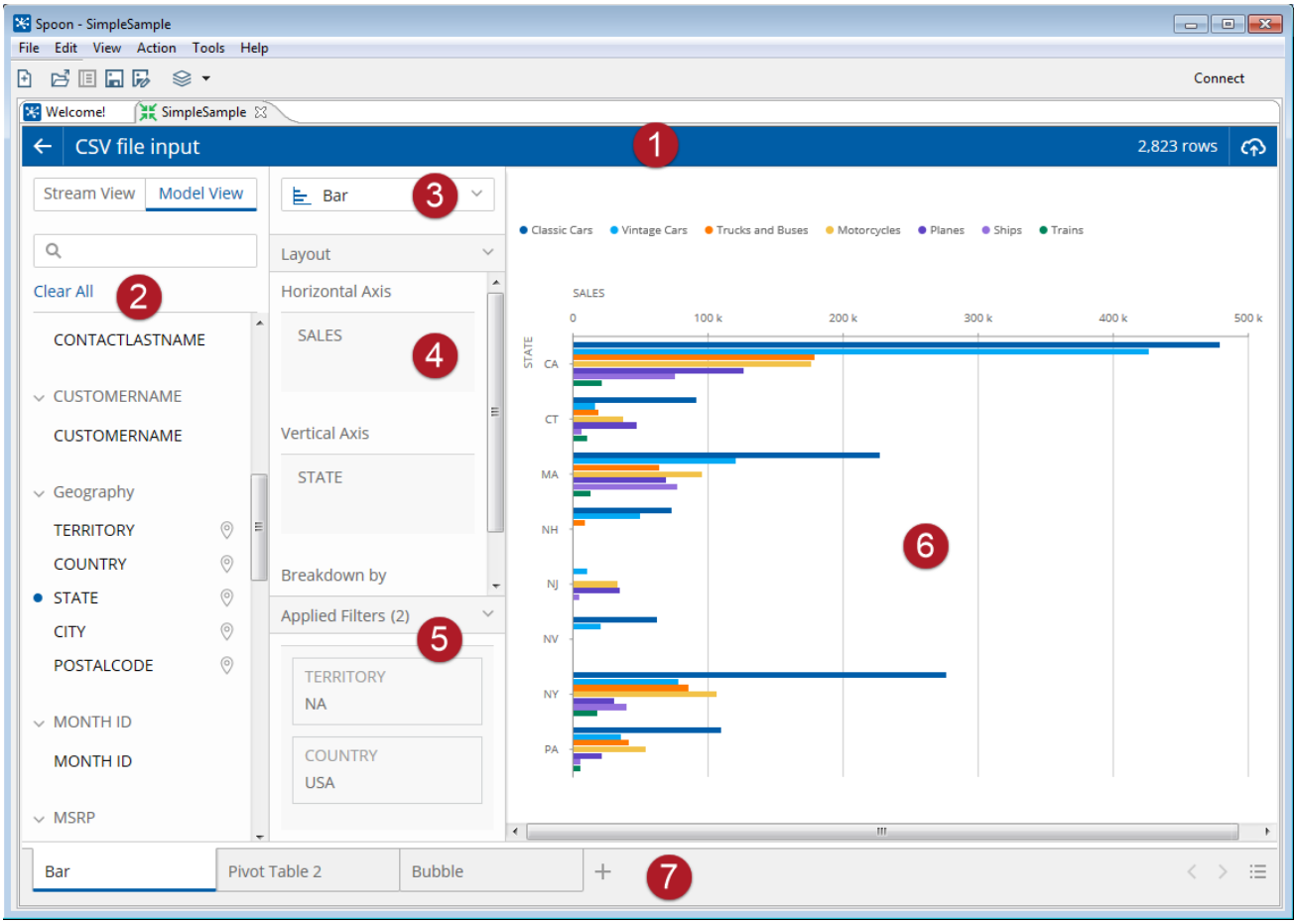
Tour the Environment

When you decide to inspect your data, the transformation presents options to visualize your data.

By default, table data is displayed with all available fields selected in **Stream View**.



The following sample screen shows a visualization using data field values from the default Stream View for a step.



This is great opportunity to involve your data consumers in how the data is to visualized.

For ETL developers, check for data consistency, validity and conformity.

Key	Name	Description
1	**Header bar**	Use the Header bar to access: • The title of the step being inspected. • The row count of the data sampled. • The **Publish** button, used to create a data source for collaborative use later via a data service. • The **Exit** button, to return to the transformation canvas
2	**Stream View / Model View**	Toggle between **Stream View** and **Model View** to inspect data and build visualizations based on the data sampled. • Use **Stream View** to inspect the data using a flat table or visualization types that do not require modeling. • **Model View** extends the analytic capabilities by allowing you to view your selected fields with hierarchical capabilities.
	Search Box	Use the **Search Box** to find a specific field in a long list of available fields. This is especially useful in Stream View where the order of the fields is determined by the transformation.
	Available Fields Panel	The **Available Fields Panel** lists all available fields from the subset of data being inspected and allows you to select the specific fields you want to inspect. Click a field to select or clear it. You can also select a field by dragging it into the **Layout Panel**. Selected fields display with a blue disk icon to the left of their names. • Use **Clear All** to remove all fields from the **Layout** panel. The **Canvas** area will be automatically updated. • For the flat table in Stream View, you can click **Select All** to include all fields in the flat table in the order they are listed.
3	**Visualization Selector**	Use the **Visualization Selector** to choose a visualization type. Selecting a visualization from the drop-down menu displays it in the **Canvas** area.
4	**Layout Panel**	Displays the properties associated with a selected category or field.
5	**Applied Filters**	In **Model View**, the **Applied Filters** panel displays all the filters you have applied to a visualization. To apply a filter, double-click an available drill-down field on the visualization. • Drill-down fields are only available in the Model View since you can only drill down in hierarchies. To drill down, double-click in a visualization data point, such as a column in a graph or a slice in a pie chart, or on a member label. • When drilling down, the next field in the hierarchy replaces its parent in the Layout panel, except in the pivot table and sunburst chart where the next field is added. The filter for the drill-down field is added to the Applied Filters panel and the visualization is updated according to those filters.
6	**Canvas**	The **Canvas** displays the selected visualization.
7	**Visualization Tabs**	Use the **Visualization Tabs** to compare multiple views of your step data.

Explore with Visualizations

When you begin inspecting your data, you are presented with the **Stream View**, with all available data fields selected. The selected data fields are represented in the **Canvas** area by a flat table. To reduce the number of data fields selected, click anywhere on a data field name. The blue dot to the left of the data field name will disappear, indicating that it is no longer selected. In some cases, it may be faster to deselect all data fields first, by clicking the Clear All actions first, then select only the data fields you want to inspect. Your selections will be listed in the order that they are selected.

Once you have the desired data fields selected, you can change the table to a different visualization type by using the **Visualization Selector**. Alternately, you can create a new visualization by clicking the plus symbol button located to the right of the current tab. Once you have a new visualization created, switch to **Model View** to display a multidimensional representation of your selected fields. If you selected a visualization that requires a multidimensional model, it will automatically switch to Model View. The Model View allows you to customize the layout, based on placement of the data fields shelves.

Publish for Collaboration

When you're ready to make your content available for others, publish it as a data source. The data source will use a data service that is automatically created on the step, which can be used by other tools later.

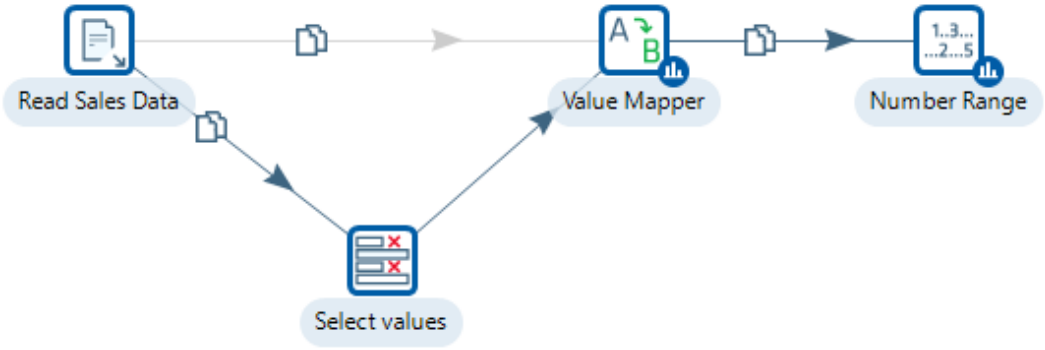
To publish, perform the following steps:

1. Click the Publish button at the top right of the Header bar. The Publish Data Source window opens.
2. Click Get Started to open the Publish Details window.
3. Enter the data source information in the following fields:

Fields	Description
Data Source Name	The name used by other Pentaho applications when accessing your data source.
Server	The default value for this field is your current repository. You can select other repository connections if you have created them through the **Repository Manager**.
URL	The base URL string used to connect to the server.
User Name	The user name required to access the server. The user must also have publish permissions.
Password	The password associated with the provided user name

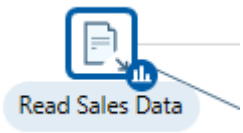
4. When you are done, click **Finish**.
5. Once your data source is created a confirmation will appear. Click **Close** to continue inspecting or return to **PUC** to work with the data source. The data source should now be available on the server.

Guided Demo: Tab Persistence

Introduction	When working with your transformation, you can gain valuable insights into the data of most steps through visualizing and interacting with your data in many ways. The demonstration illustrates the concept of Tab Persistence; the state of the visualization is 'saved' – persisted – with the transformation.
Objectives	In this guided demonstration, you will… • Explore the data • Visualize the data • Set Tab Persistence
Transformation	

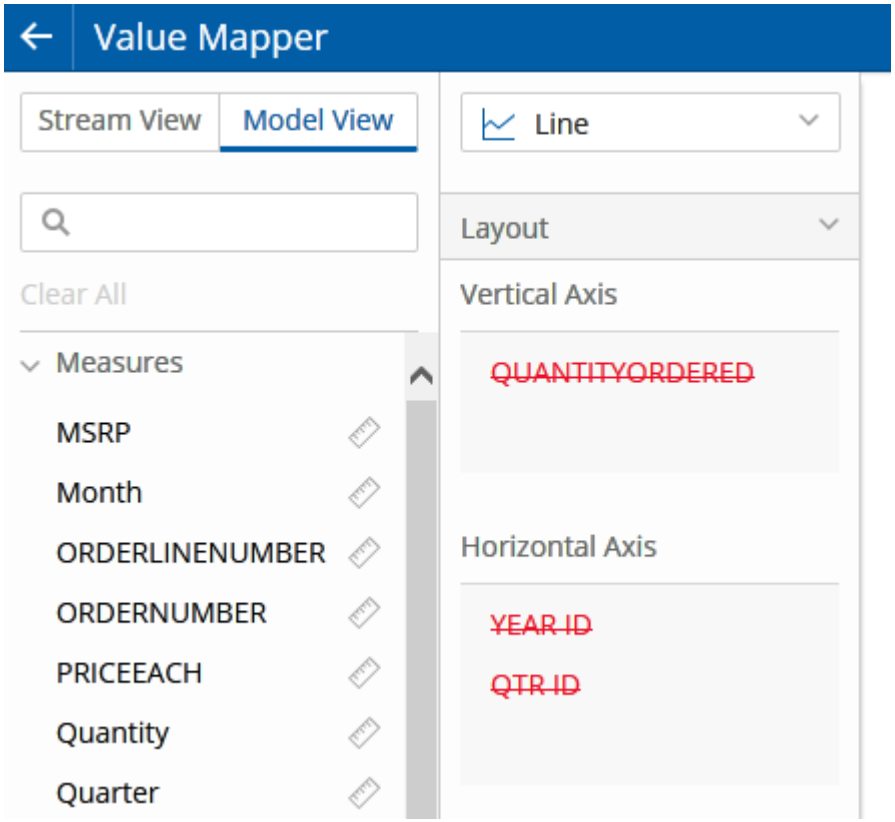
Persistence Tabs

1. Inspect the Read Sales Data step and create a visualization in Data Explorer; an icon appears - it persists the state of the step automatically.



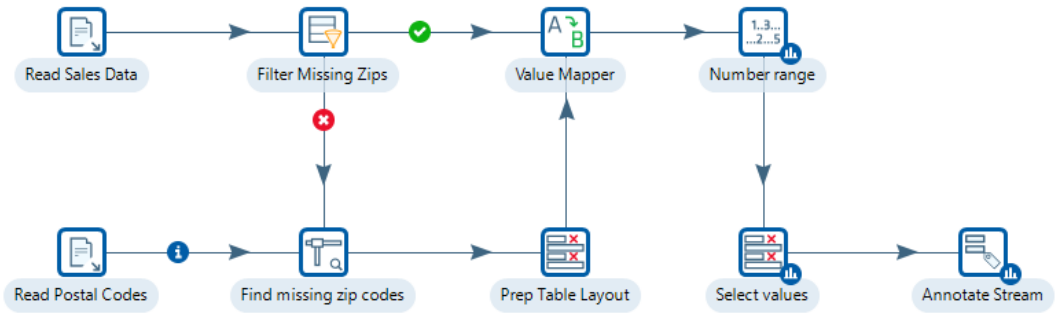
   Ensure that the Hop between Read Sales Data and Value mapper is disabled.

2. Click on the Value Mapper step, then 'Inspect Data' in Data Explorer. Notice that as the values have been removed or changed, as a result the visualization wont render.



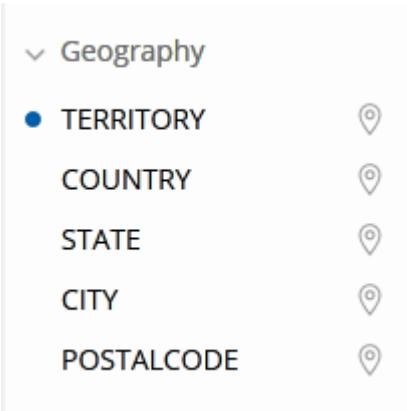
3. Enable the Hop between the Read Sales Data and Value Mapper, and disable the Hops between Read Sales Data > Select values > Value mapper.
4. Select the Value Mapper and 'Inspect Data', in Data Explorer. The visualization successfully renders.
5. In the Number Range step, view the visualization. These are based on the number bands set in the step.
6. Change the values in the step and notice the changes.
7. Add a new table tab and then exit. When you next open Data Explorer, focus will remain on the table tab (the last tab viewed).
8. Switch your first visualization on the CSV input tab to the table view. When you exit the step, the content icon will disappear.

Guided Demo: Visualizations & Drilldown

Introduction	When working with your transformation, you can gain valuable insights into the data of most steps through visualizing and interacting with your data in many ways. The demonstration illustrates how 'Drilldown' is resolved in the Model view.
Objectives	In this guided demonstration, you will... • Explore the data • Visualize the data • Drilldown
Transformation	

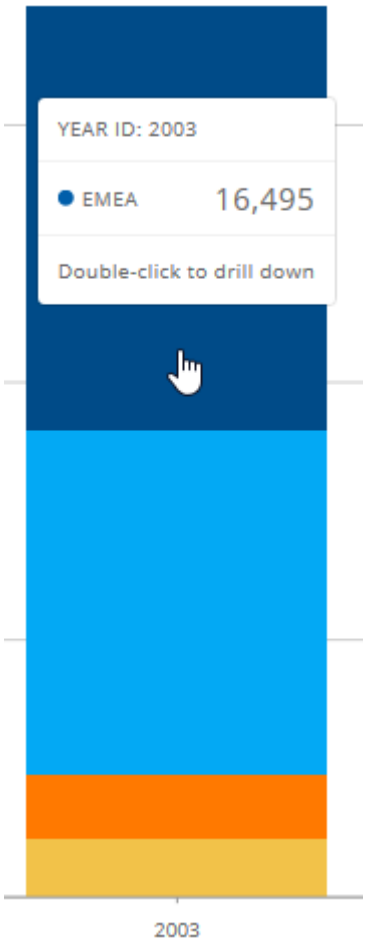
Drilldown

1. Click on the 'Number Range' step; then 'Inspect Data' in Data Explorer. The model will attempt to resolve measures, dimensions, and geo hierarchies automatically.

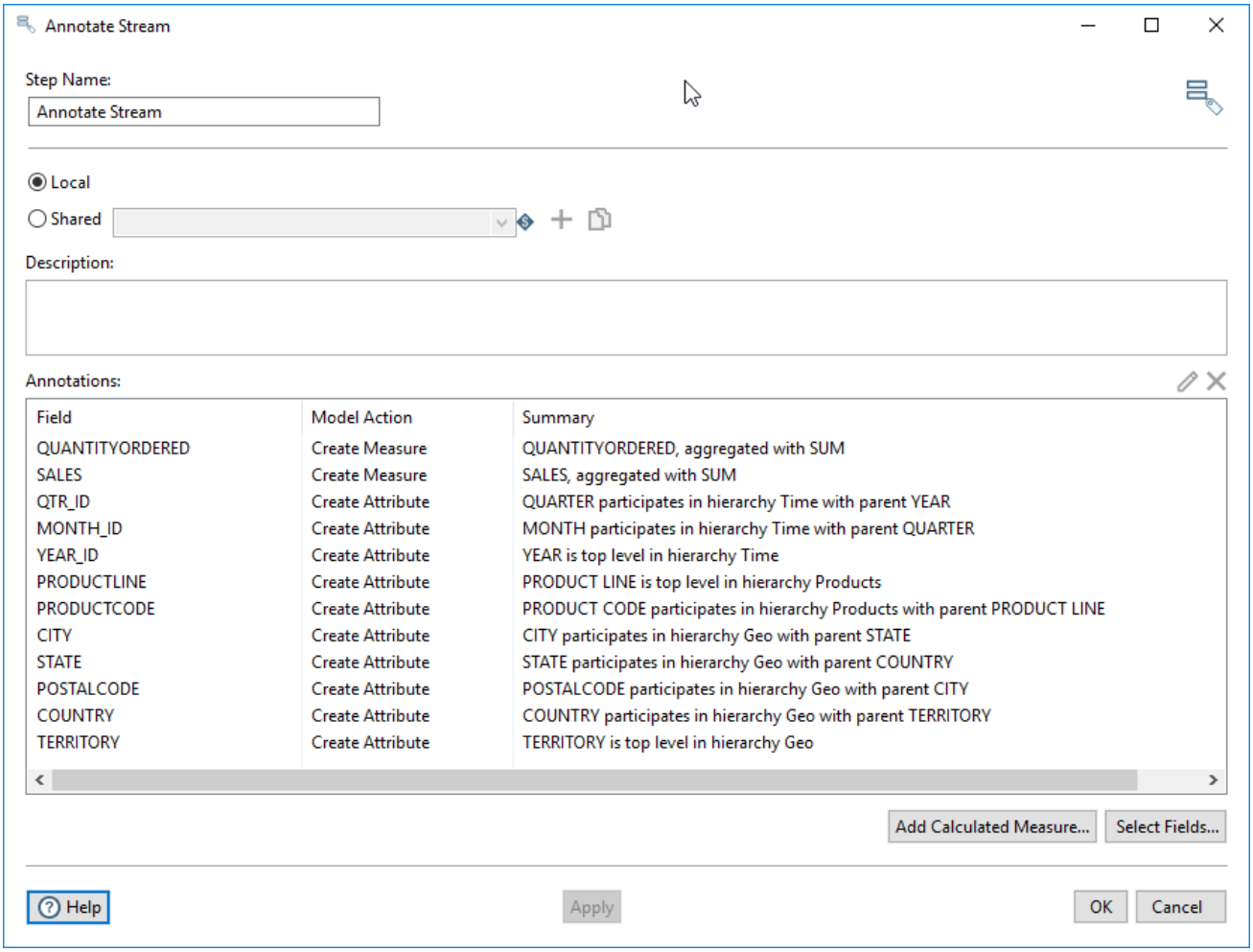


Note the new search box to find fields.

2. Click on the Select values step. Double – click on the stacked bar chart and drilldown from Territory to Country.

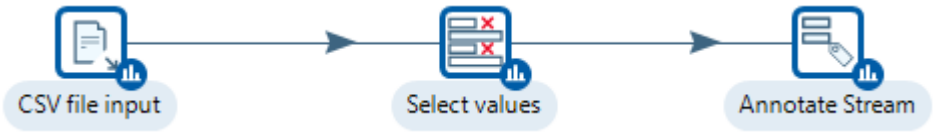


3. The Annotate stream step improves the auto modelling results. The step adds a time hierarchy, product hierarchy, specified measure formats/aggregations, and more.



4. Note order of fields drilled on when you double click on the visualizations in Annotate Stream:
   - Bar/Column chart: drills first on breakdown field, then axis
   - Scatter: drills first on colour field, then dots
   - Heat grid: Drills first on horizontal field, then vertical
   - Sunburst: Drills based on slice you click

Guided Demo: Geo Capabilities

Introduction	When working with your transformation, you can gain valuable insights into the data of most steps through visualizing and interacting with your data in many ways. The demonstration illustrates the concept of Geo Capabilities; Longitude and Latitude coordinates need to be associated with the correct attribute.
Objectives	In this guided demonstration, you will… • Explore the data • Visualize the data • Geo capabilities
Transformation	

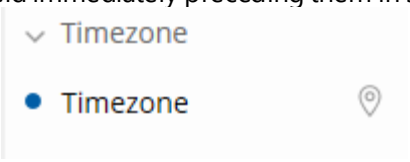
CSV File Input

Data on Starbucks store locations in USA and Canada; fictitious sales numbers. The auto-model in Data Explorer recognizes Country and City. The model view displays incorrect latitude and longitude, as they're associated with the incorrect field (Timezone).

1. Create a Geo Map report:
   - Geography – Timezone
   - Color - Stores
   - Size – Est.Sales

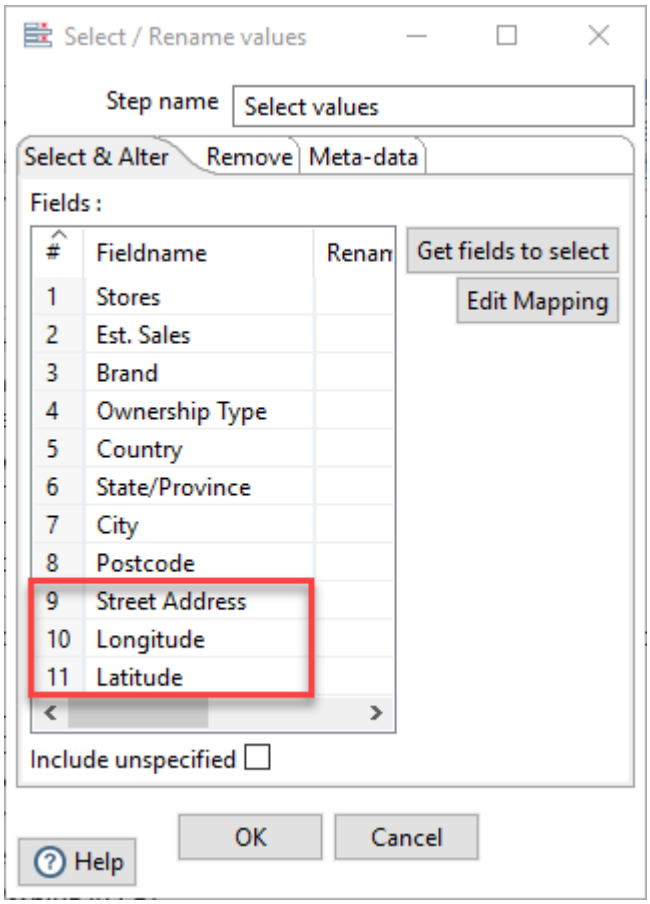
Auto model associates latitude and longitude fields to the field immediately preceding them in stream.

11	Timezone	String
12	Longitude	Number
13	Latitude	Number

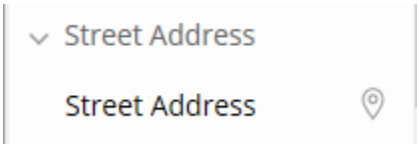


Select Values

1. Reorder the fields in 'Select Values' so street address precedes latitude and longitude



Auto model now associates latitude and longitude to Street Address.



2. Create a Geo Map report:
   - Geography – City, Street Address
   - Color - Stores
   - Size – Est.Sales

   Zoom in to the street..

Note that if you pan/zoom in the map, and then close Data Explorer, your focus area on the map is not persisted.

Guided Demo: Performance

Introduction	When working with your transformation, you can gain valuable insights into the data of most steps through visualizing and interacting with your data in many ways. The demonstration illustrates the concept of Tab Persistence; the state of the visualization is 'saved' – persisted – with the transformation.
Objectives	In this guided demonstration, you will… • Explore the data • Visualize the data • Set Tab Persistence

CSV File Input

This is a file with over 240,000 rows of data.

Data Explorer v7.1+ loads rows incrementally, so the user no longer needs to wait until all rows are loaded to see data. When you hit 'inspect', try to scroll down -- you should see a 'processing' message once you have reached the end of rows loaded so far.

You will also notice that after opening Data Explorer once by hitting 'inspect,' subsequent opens will take less time. This is because the Data Explorer browser remains open in background.

The Data Explorer row limit can be increased above 50,000:

1. Launch PDI
2. From the tool bar: Edit > Edit kettle.properties
3. Scroll down and 'Insert' a new row
4. For the "variable name" type: det.dataservice.dynamic.limit
5. For the "value" field type: [any number different than 50000]
6. Restart Spoon

**Please note that increasing this limit can lead to performance issues.

Summary

Topics covered in this section:

- Transformations
- Parallelism
- Rows of Data
- Data Types
- Data Explorer

Pentaho Data Integration prepares and blends data to create a complete picture of your business that drives actionable insights. The platform delivers accurate, analytics-ready data to end users from any source. With visual tools to eliminate coding and complexity, Pentaho puts big data and all data sources at the fingertips of business and IT users.

Intuitive drag-and-drop data integration coupled with data agnostic connectivity spanning from flat files and RDBMS to Hadoop and beyond.

- Graphical extract-transform-load (ETL) designer to simplify the creation of data pipelines
- Rich library of pre-built components to access, prepare, and blend data from relational sources, big data stores, enterprise applications, and more
- Powerful orchestration capabilities to coordinate and combine transformations, including notifications and alerts
- Agile views for modelling and visualizing data on the fly during the data preparation process
- Integrated enterprise scheduler for coordinating workflows and debugger for testing and tuning job execution

You also configured the following key Transformation steps:

- CSV File Input - This step provides the ability to read data from a delimited file. The CSV label for this step is a misnomer because you can define whatever separator you want to use, such as pipes, tabs, and semicolons; you are not constrained to using commas. Internal processing allows this step to process data quickly. Options for this step are a subset of the Text File Input step.
- Dummy - The Dummy step does not do anything. Its primary function is to be a placeholder for testing purposes.
- Generate rows - Generate rows outputs a specified number of rows. By default, the rows are empty; however, they can contain several static fields. This step is used primarily for testing purposes. It may be useful for generating a fixed number of rows, for example, you want exactly 12 rows for 12 months. Sometimes you may use Generate Rows to generate one row that is an initiating point for your transformation. For example, you might generate one row that contains two or three field values that you might use to parameterize your SQL and then generate the real rows.
- Text File Output - is used to export data to text file format. This is commonly used to generate Comma Separated Values (CSV files) that can be read by spreadsheet applications. It is also possible to generate fixed width files by setting lengths on the fields in the fields tab.

You also configured the following key Job Entries:

- START - defines the starting point for job execution. Every job must have one (and only one) Start. Unconditional job hops only are available from a Start job entry.
- Transformation - is used to execute a previously defined transformation.
- Success - This step clears any error state encountered in a job and forces it to a success state.

In the next Module, we look at Flat files as a Datasource.

3: Data Sources: Working with Files

Topics covered in this section:

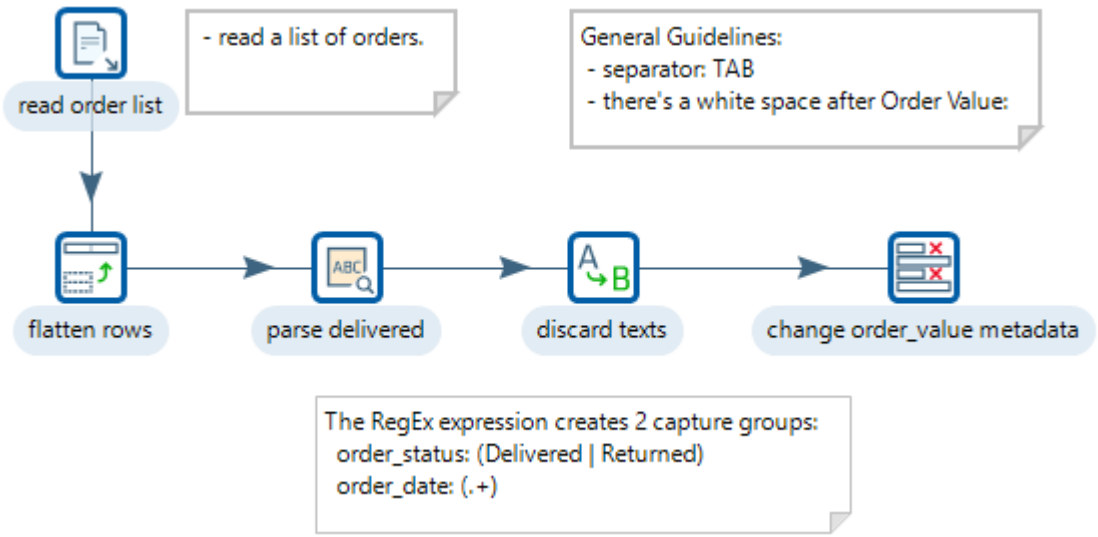
- Text file as a data source:
  - Text File Input
  - Text File Output
  - Excel Writer
  - Get data from XML
  - JSON

Despite being the most basic format used to store data, files are broadly used and they exist in several formats as fixed width, comma-separated values, spreadsheet, or even free format files. Pentaho Data Integration (PDI) can read data from all types of files; in this section, you will retrieve and write data in a variety formats with PDI.

Guided Demo: Text File Input

Introduction	Some of the Orders data that Steel Wheels process are in a text format. In this guided demo, you will flatten the list, create capture groups, replace text, and finally format the order_value.

Objectives	In this guided demonstration, you will:
	• Configure the following steps:
	  ○ Text File Input
	  ○ Flattener
	  ○ RegEx Evaluation
	  ○ Replace in String
	  ○ Select values

Transformation	

To create a new transformation:

1. In Spoon, click File > New > Transformation.

Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

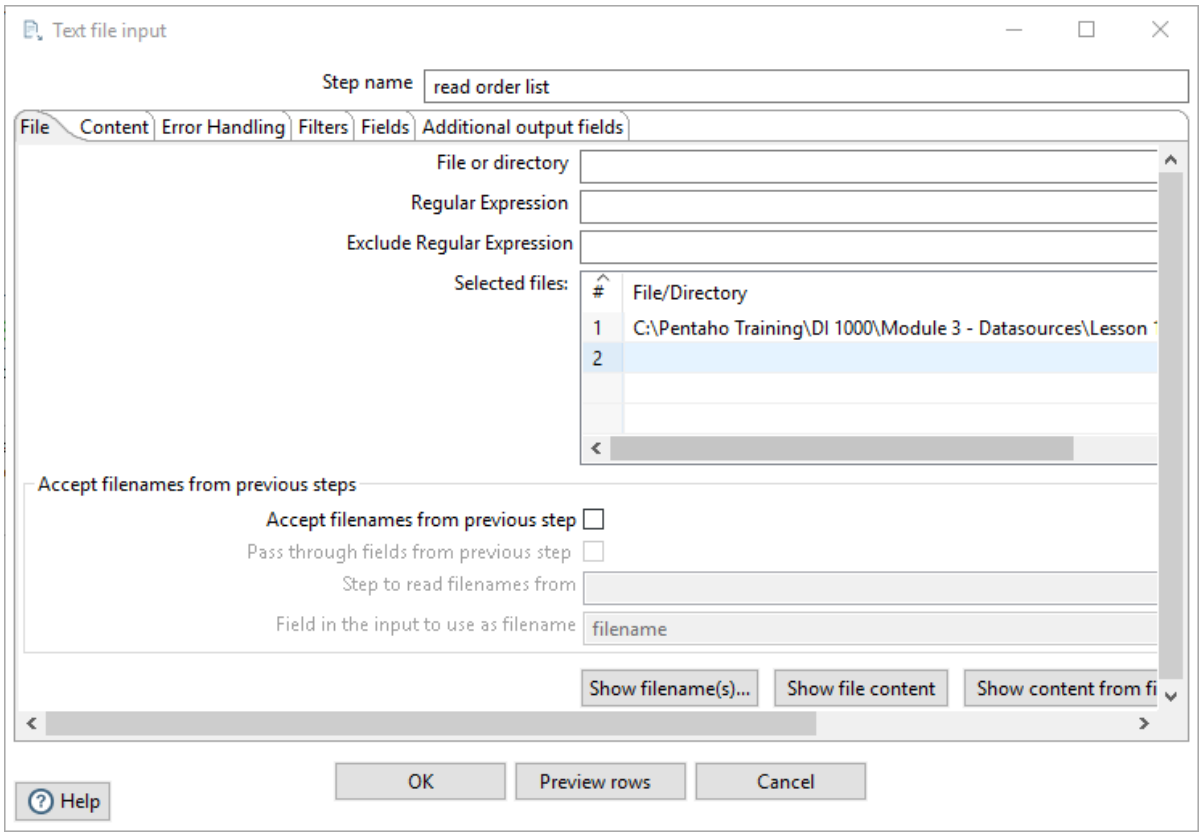
- By clicking New, then Transformation
- By using the CTRL-N hot key

Text File Input – Read Order List

The Text File Input step is used to read data from a variety of different text-file types. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

The Text File Input step provides you with the ability to specify a list of files to read, or a list of directories with wild cards in the form of regular expressions. In addition, you can accept filenames from a previous step making filename handling more even more generic.

1. Drag the 'Text File Input' step onto the canvas.
2. Double-click on the step, and configure the following properties:
   File: C:\Pentaho Training\DI 1000\Module 3 - Datasources\Lesson 1 - Working with Files\Demo 3-1-1 - Read Text File\orders.txt

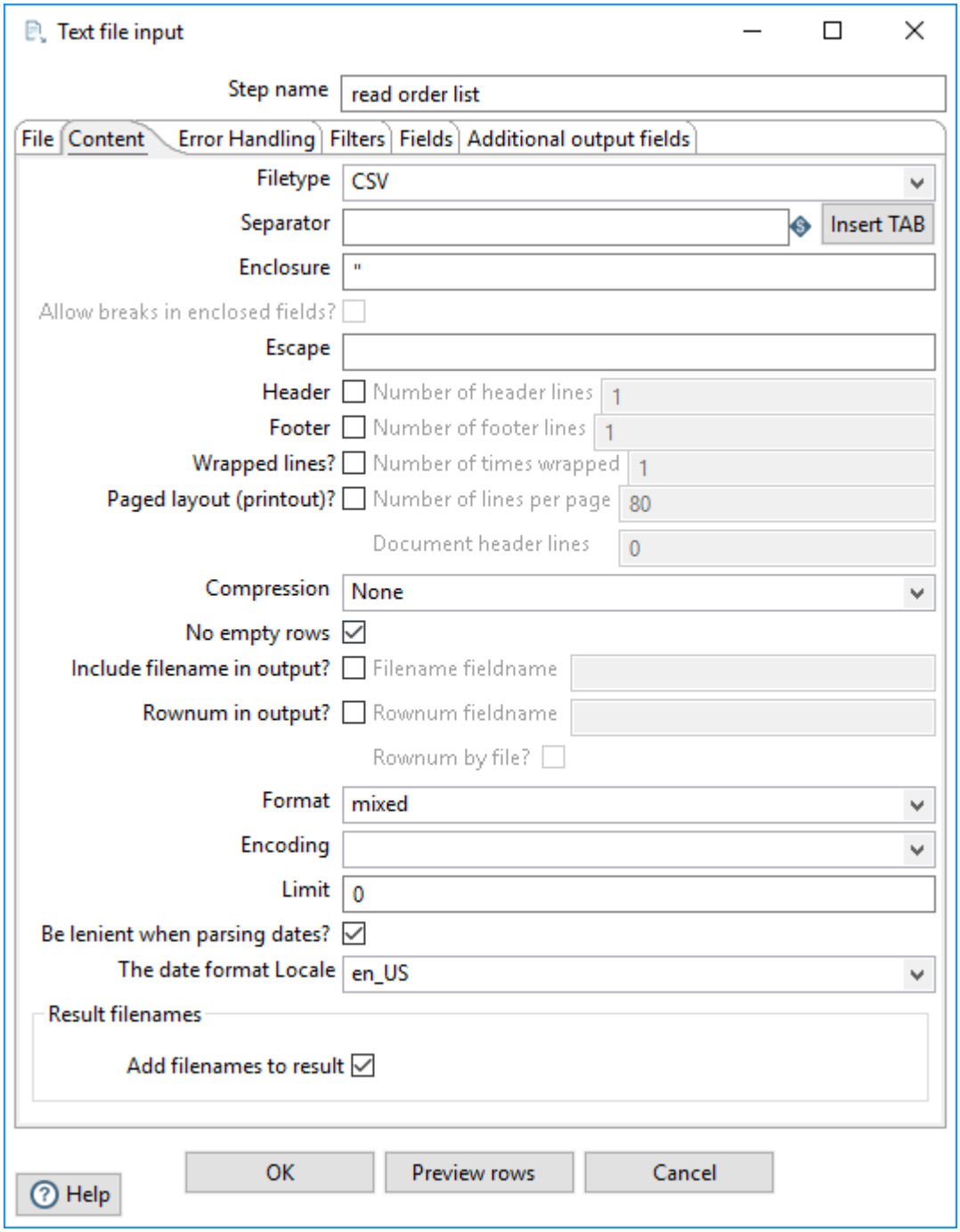


- You may find some of the paths to files set using internal variables.

  Because the sample file is located in the same directory where the transformation resides, a good approach to naming the file in a way that is location independent is to use a system variable to parameterize the directory name where the file is located. In our case, the complete filename is:

  ${Internal.Transformation.Filename.Directory}/orders.txt

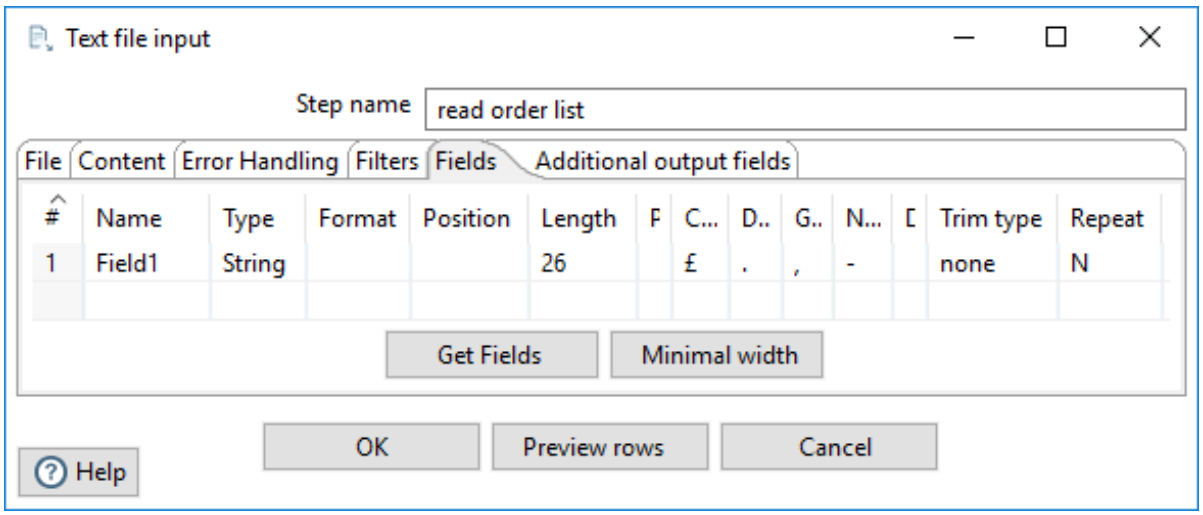
3. Click on the 'Content' tab and configure the following properties:



- In our example the separator is a TAB.
- To highlight the TAB Insert, place your cursor in the field , hold down the left mouse button and drag across the filed.

One or more characters that separate the fields in a single line of text. Typically this is ; or a TAB. Special characters (e.g. CHAR ASCII HEX01) can be set with the format $[value], e.g. $[01] or $[6F,FF,00,1F]

4. Click on the 'Fields' tab and configure the following properties:
   - Click on 'Get Fields' button.



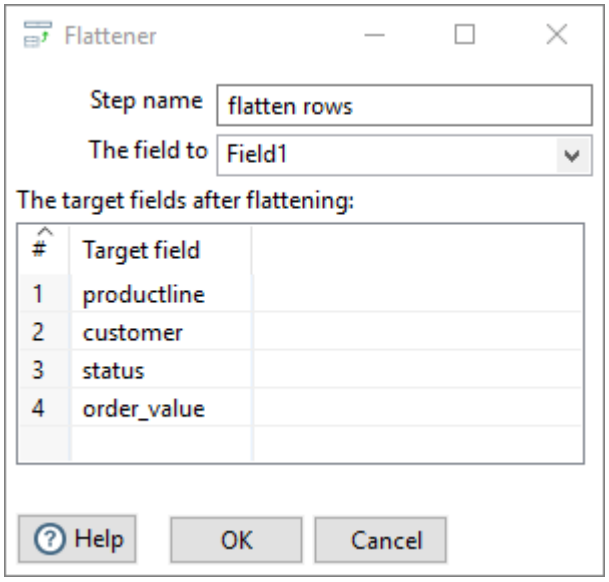
5. Close the step.

- The dataset is associated with 'Field1' with a data type of String, in the data stream.

Flattener - Flatten Rows

The Flattener step allows you flatten data sequentially.

1. Drag the 'Flattener' step onto the canvas.
2. Create a hop from the read order list step.
3. Double-click on the step, and configure the following properties:



4. Close the step.

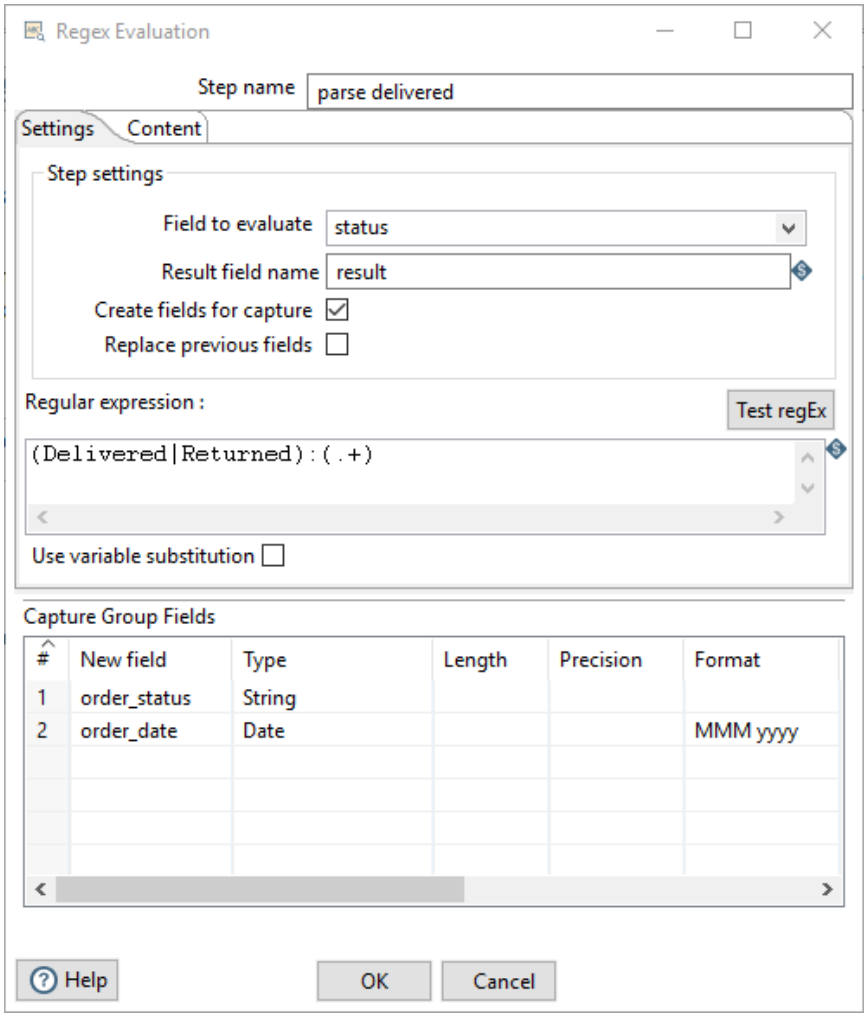
- The data has now been flattened into records. This step enables you to define new target fields that match the number of repeating records. So Target field 1 will map to repeating record 1, and so on..

RegEx Evaluation – Parse Delivered

This step type allows you to match the String value of an input field against a text pattern defined by a regular expression. Optionally, you can use the regular expression step to extract substrings from the input text field matching a portion of the text pattern into new output fields. This is known as "capturing".

In our example, we're going to extract and create two capture groups order_status and order_date based on the regex expression: (Delivered|Returned):(.+)

1. Drag the 'RegEx Evaluation' step on to the canvas.
2. Create a hop from the 'flatten rows' step.
3. Double-click on the step, and configure the following properties:



You will also need to set Trim: both for each field. This will ensure all white space is removed and the exact length of the field is returned.

4. Close the step.

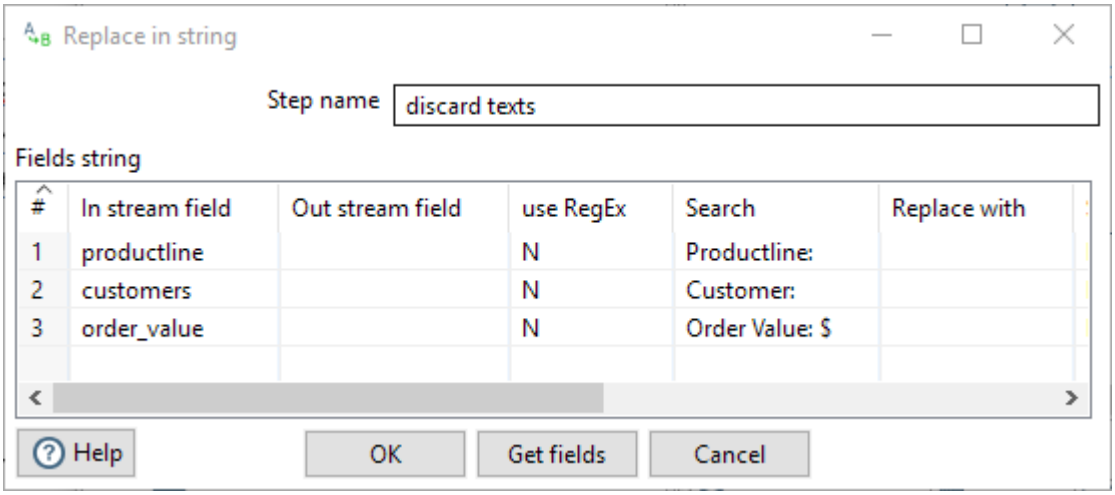
- This RegEx uses 2 constructs, denoted by the brackets, and separated by a full colon.
- (Delivered | Returned) – match against Delivered or Returned.
- (.+) matches any character
- You can Test regEx to see if the capture groups are correctly defined.
- A good introduction can be found at: https://regex101.com/

Replace in String – discard texts

Replace in string is a simple search and replace. It also supports regular expressions and group references. Group references are picked up in the replace by string as $n where n is the number of the group.

Time to tidy up the order_value stream field data. In this step, you replace the Order Value: with 'nothing'.

1. Drag the 'Replace in String' step onto the canvas.
2. Create a hop from the 'parse delivered' step.
3. Double-click on the step, and configure the following properties:



4. Close the step.

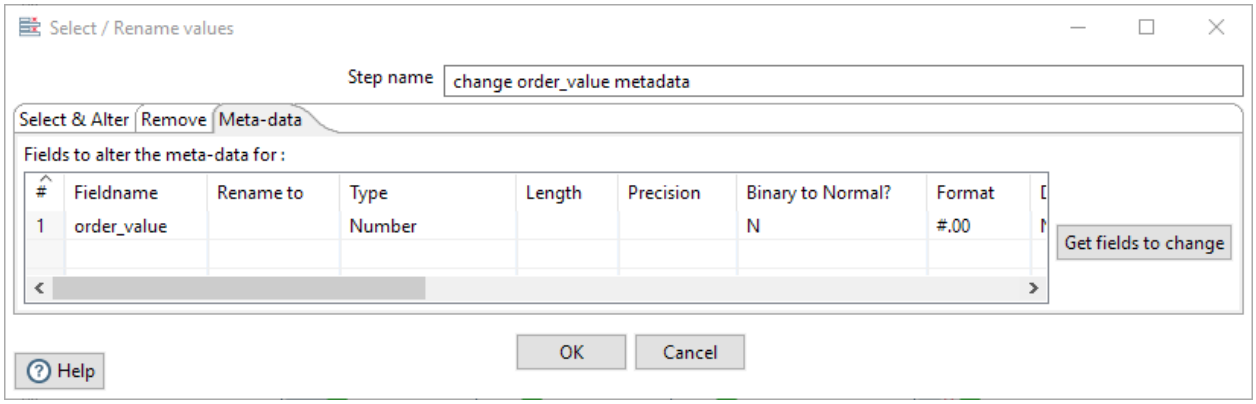
- Ensure you have correctly entered the Search: Order Value: $

Select Values – change order_value metadata

The Select Values step is useful for selecting, removing, renaming, changing data types and configuring the length and precision of the fields on the stream. These operations are organized into different categories:

- Select and Alter — Specify the exact order and name in which the fields should be placed in the output rows
- Remove — Specify the fields that should be removed from the output rows
- Meta-data - Change the name, type, length and precision (the metadata) of one or more fields

1. Drag the Select values step onto the canvas.
2. Create a hop from the 'discard texts' step.
3. Double-click on the step, and configure the following properties:



Fieldname	order_value
Type	Number
Format	#.00

4. Close the step.

RUN the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Click on the Preview tab.

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

First rows | Last rows | Off — Inspect Data

#	productline	customers	status	order_value	result	order_status	order_date
1	Classic Cars	Christine Loomis	Delivered: January 2004	21.99	Y	Delivered	Jan 2004
2	Classic Cars	Mary L. Peachin	Delivered: November 2008	24.99	Y	Delivered	Nov 2008
3	Trains	Bob Italia	Delivered: July 1994	14.99	Y	Delivered	Jul 1994
4	Planes	Scott M. Ascher	Delivered: March 2014	27.99	Y	Delivered	Mar 2014
5	Motorcycles	Monty Halls	Returned: April 2007	29.99	Y	Returned	Apr 2007
6	Trains	Paul McCallum	Returned: June 2017	34.99	Y	Returned	Jun 2017
7	Boats	Jill Robinson	Delivered: November 2014	19.99	Y	Delivered	Nov 2014

Guided Demo 3-1-2: Text File Output

Introduction

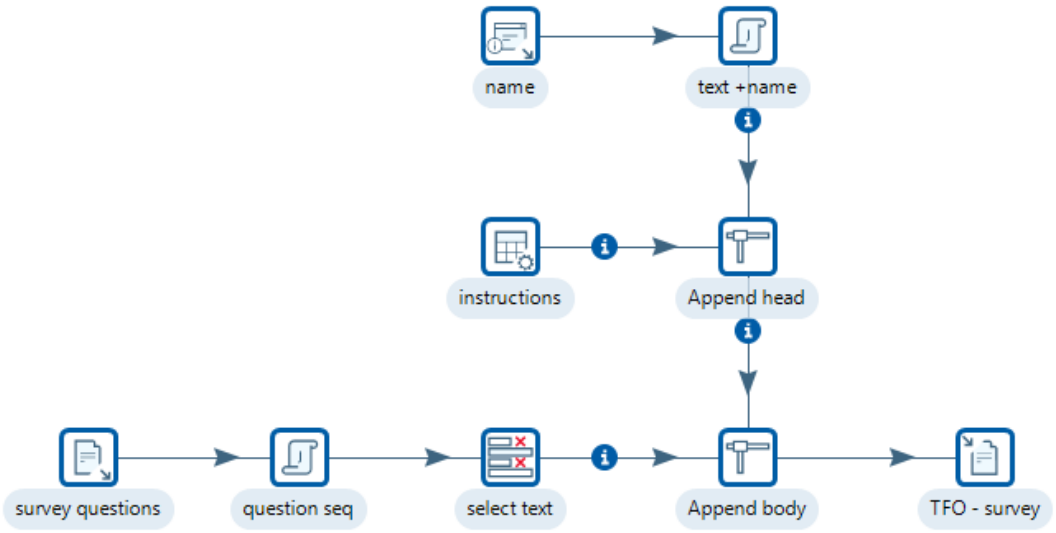
Steel Wheels wants to send out a survey to its customers, based on a list of questions.

Objectives

In this guided demonstration, you will:

- Configure the following steps:
  - Get System Info
  - User Defined Java Expression
  - Data Grid
  - Append
  - Text File Output

Transformation



To create a new transformation:

1. In Spoon, click File > New > Transformation.

Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

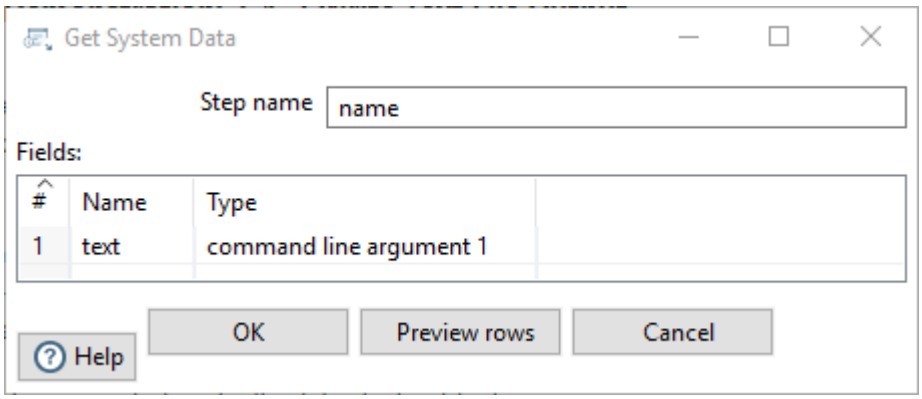
- By clicking New, then Transformation
- By using the CTRL-N hot key

Get System Info - name

The Get System Info step retrieves information from the Kettle environment. This step generates a single row with the fields containing the requested information. It also accepts input rows. The selected values are added to the rows found in the input stream(s).

We will use this step to input the Customer Name as an argument.

1. Drag the 'Get System Info' onto the canvas.
2. Double-click on the step, and configure the following properties:



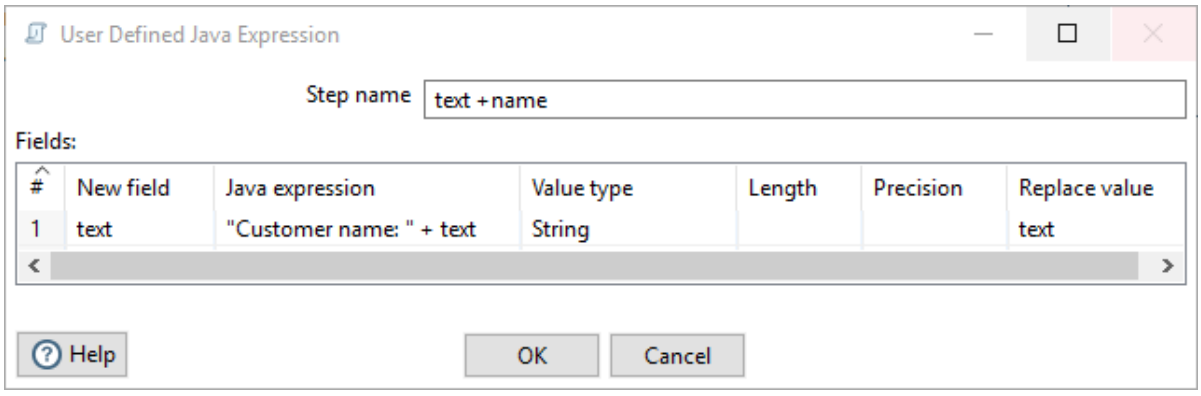
3. Close the step.

- The name of the customer is now associated with the stream field 'text'.

User Defined Java Expression - text + name

This step allows you to enter User Defined Java Expressions as a basis for the calculation of new values. In this example, a user defined java expression is used to update the 'text' stream field with the Customer Name.

1. Drag the 'User Defined Java Expression' onto the canvas.
2. Create a hop from the 'name' step.
3. Double-click the step, and configure the following properties:



4. Close the step.

- Replaces the previous 'text' stream field (Argument value) with concatenation of Customer name + Argument value.
- Quick Reference : `https://docs.oracle.com/javase/tutorial/java/nutsandbolts/opsummary.html`

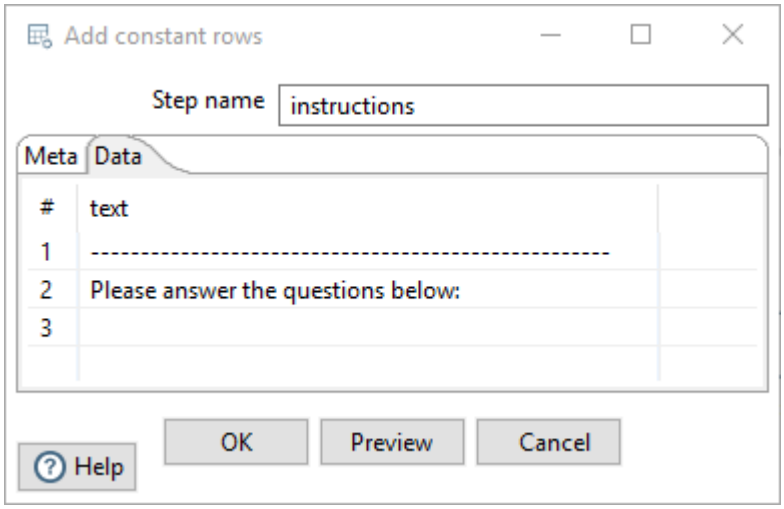
Data Grid (Add Constant Rows) - instructions

The Data Grid step allows you to enter a static list of rows in a grid. This is usually done for testing, reference or demo purposes.

- Meta tab: on this tab, you can specify the field metadata (output specification) of the data
- Data tab: This grid contains the data. Everything is entered in String format so make sure you use the correct format masks in the metadata tab.

We're going to use this step to define the top section - head – of the survey.

1. Drag a 'Data Grid' step onto the canvas.
2. Double-click the step, and configure the following properties:



3. Close the step.

- Adds text 'layout'.

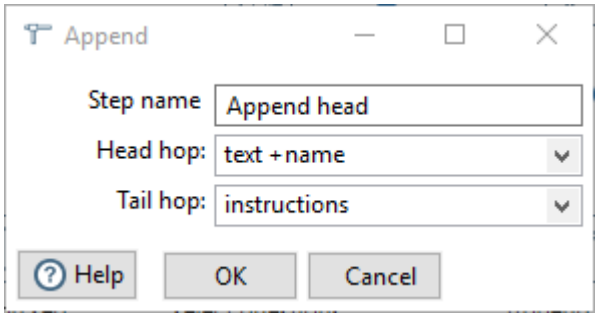
Append – Append head

This step type allows you to order the rows of two inputs hops. First, all the rows of the "Head hop" will be read and output, after that all the rows of the "Tail hop" will be written to the output. If more than 2 hops need to be used, you can use multiple append steps in sequence.

As always, the row layout for the input data coming from both steps must be identical: the same row lengths, the same data types, the same fields at the same field indexes in the row.

In our example, the Head hop 'text + name' is appended to the Tail hop, 'questions'.

1. Drag the 'Append' step onto the canvas.
2. Create hops from the 'text + name' and 'questions' steps.
3. Double-click on the step, and configure the following properties:



4. Close the step.

- Appends the 'Tail hop' - text layout to the 'Head hop' - Customer name: 'name'.
- The trick here is, as the we are appending data streams to keep the stream fields and data type consistent. Each stream must have the same layout and data type: in this case 'text' and 'string'

Important: If you don't care about the order in which the output rows occur, you can use any step to create a union of 2 or more data streams.

Text File Input - survey questions

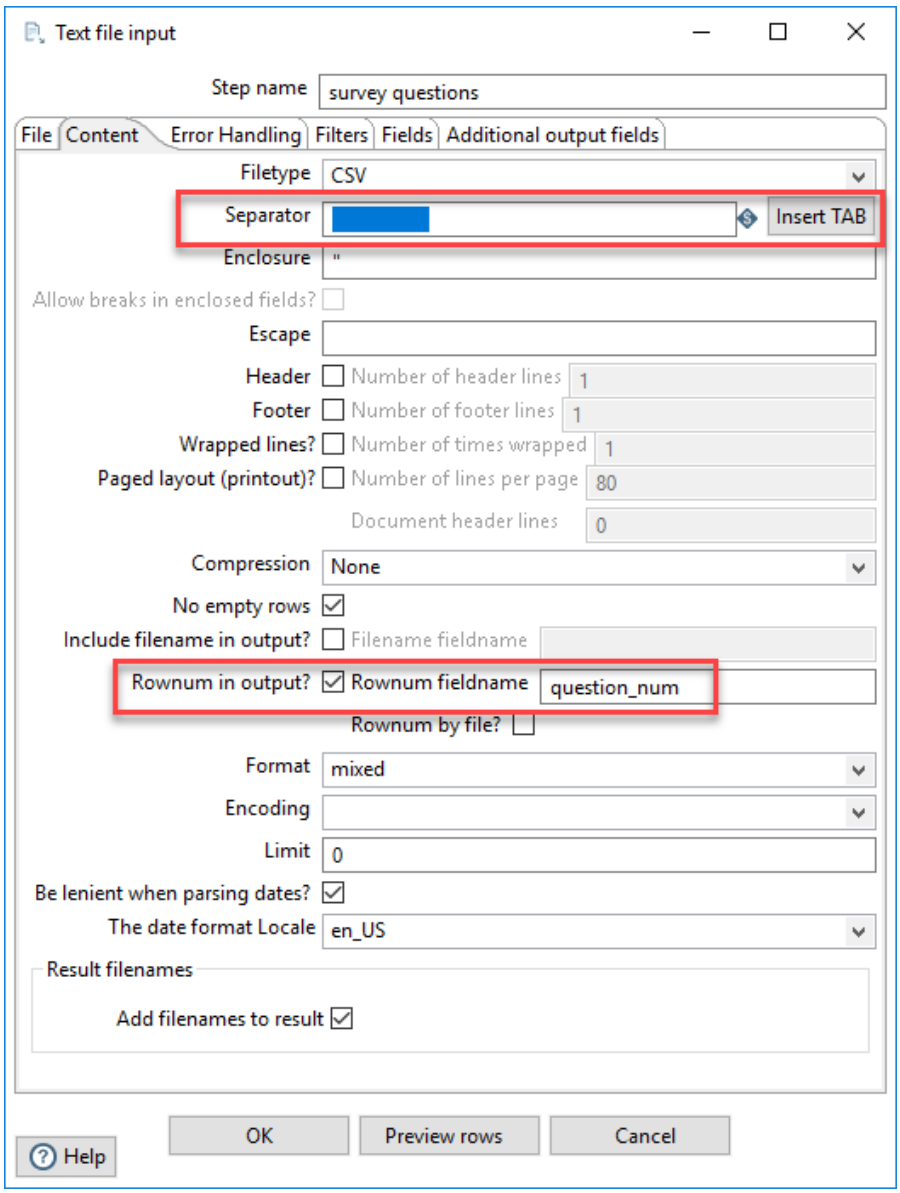
The Text File Input step is used to read data from a variety of different text-file types. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

The Text File Input step provides you with the ability to specify a list of files to read, or a list of directories with wild cards in the form of regular expressions. In addition, you can accept filenames from a previous step making filename handling more even more generic.

Part II – the main objective is to append the questions.txt to the 'head' stream.

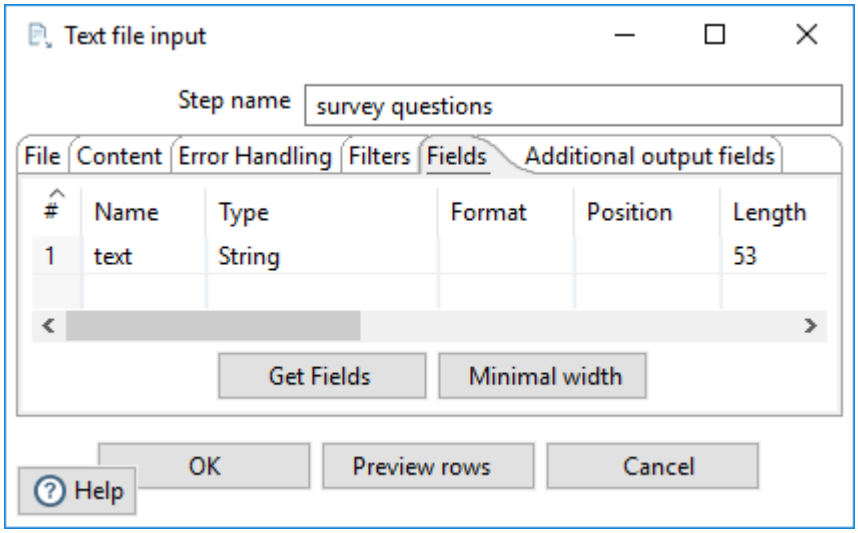
1. Drag the 'Text File Input' step onto the canvas.
2. Double-click on the step, and configure with the following properties:

   File: `C:\Pentaho Training\DI 1000\Module 3 - Datasources\Lesson 1 - Working with Files\Demo 3-1-2 - Write Text File\questions.txt`



- Uses row numbers as question numbers.

3. Rename the stream field: text



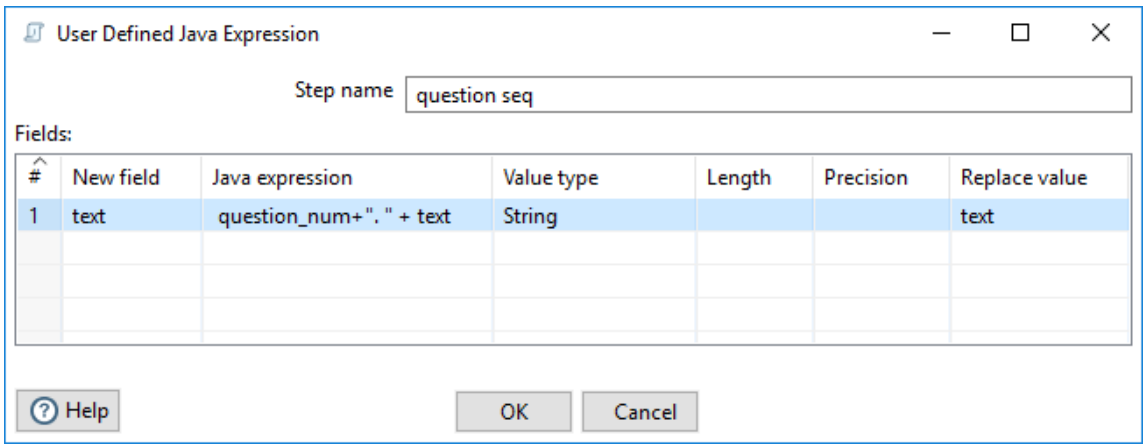
4. Close the step.

- The questions are now associated with 'text' stream field, with each record numbered by the rownum fieldname: question_num.

User Defined Java Expression – question seq

This step allows you to enter User Defined Java Expressions as a basis for the calculation of new values. In this example, a user defined java expression is used to update the 'text' stream field with the 'question_num'.

1. Drag the 'User Defined Java Expression' onto the canvas.
2. Create a hop from the 'survey questions' step.
3. Double-click the step, and configure the following properties:



4. Close the step.

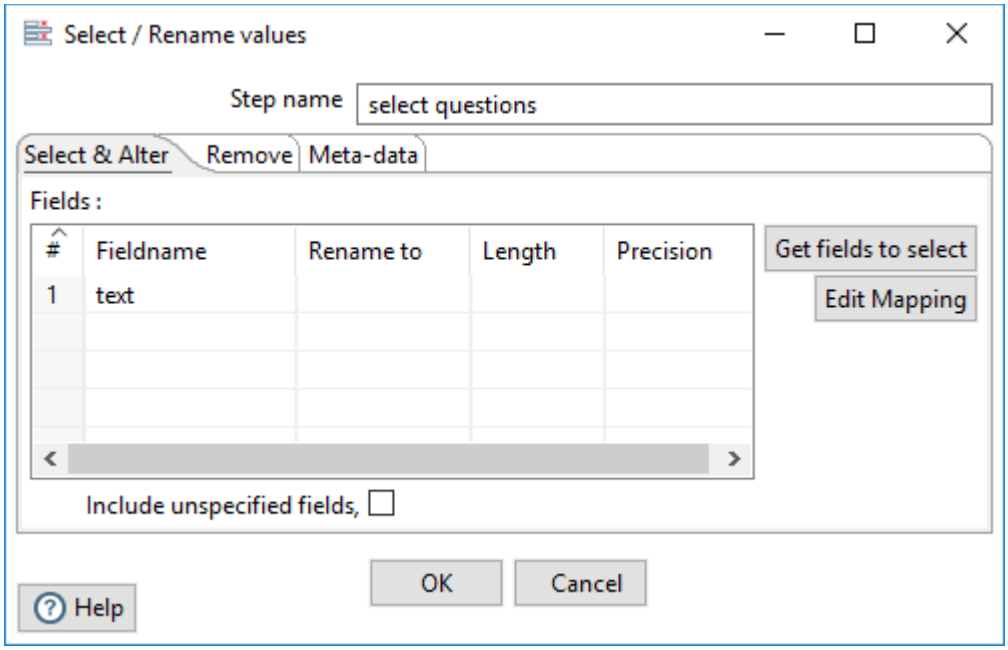
- Adds question_num (question numbers) to the 'text' stream field

Select Values - select questions

The Select Values step is useful for selecting, removing, renaming, changing data types and configuring the length and precision of the fields on the stream. These operations are organized into different categories:

- Select and Alter — Specify the exact order and name in which the fields should be placed in the output rows
- Remove — Specify the fields that should be removed from the output rows
- Meta-data - Change the name, type, length and precision (the metadata) of one or more fields

1. Drag the Select values step onto the canvas.
2. Create a hop from the 'question seq' step.
3. Double-click on the step, and configure the following properties:



4. Close the step.

- Selects just the text field

The reason why you need a 'Select values' step is to select the correct data stream field - text - that will be appended to the 'Body' stream. Currently there are two stream fields:

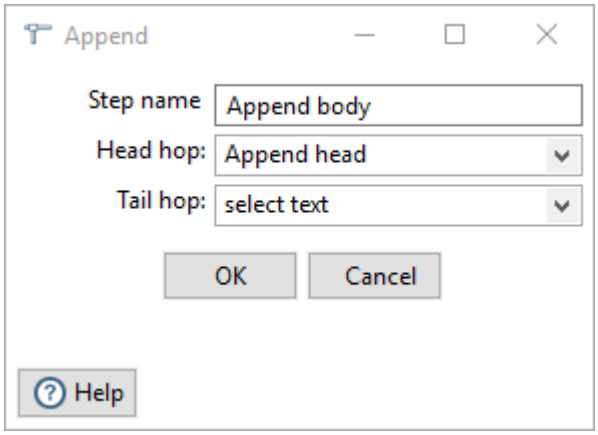
- question_num - coming from the Text File Input - survey questions
- text – the main datastream field.

Append - Append body

This step type allows you to order the rows of two inputs hops. First, all the rows of the "Head hop" will be read and output, after that all the rows of the "Tail hop" will be written to the output. If more than 2 hops need to be used, you can use multiple append steps in sequence.

In our example, the Head hop 'Append head' is appended to the Tail hop, 'select questions'.

1. Drag the 'Append Streams' step onto the canvas.
2. Create hops from the 'Append head' and 'select questions' steps.
3. Double-click on the step, and configure the following properties:



4. Close the step.

- The two data streams are now appended under one data stream field: text

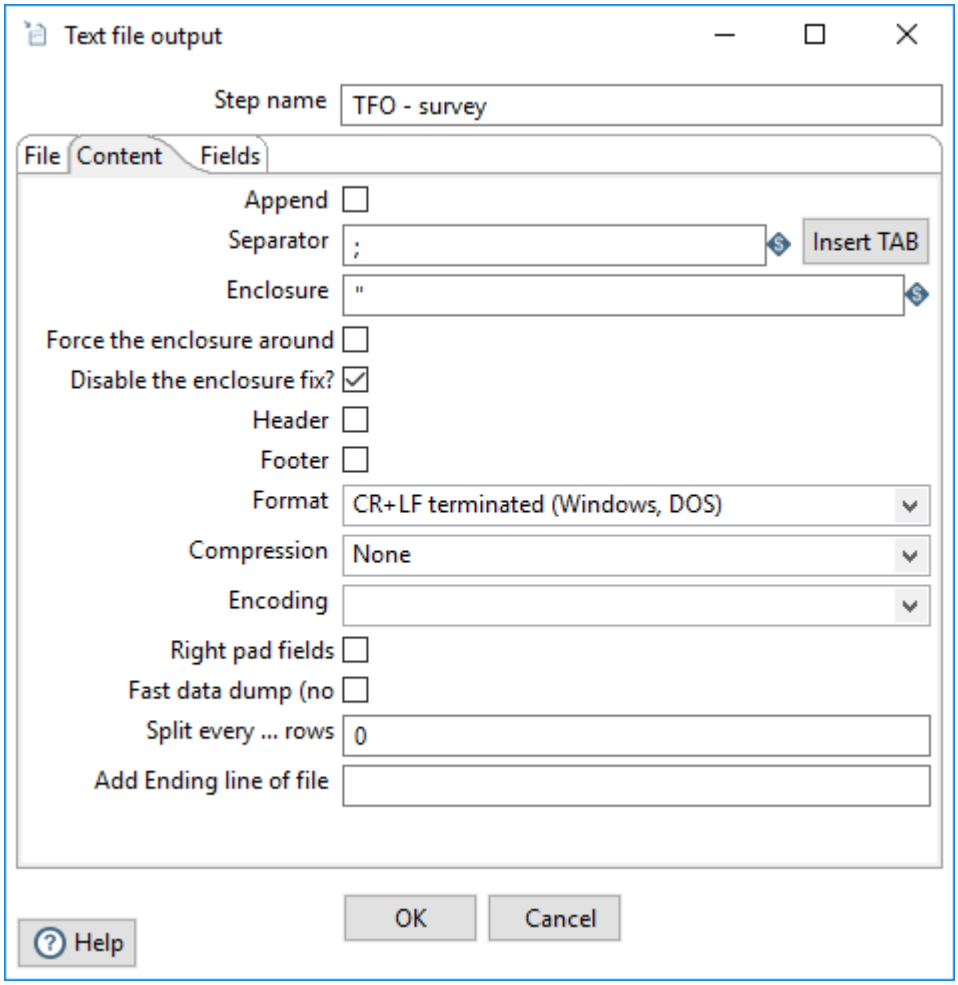
Text File Output - TFO survey

The Text file output step is used to export data to text file format. This is commonly used to generate Comma Separated Values (CSV files) that can be read by spreadsheet applications. It is also possible to generate fixed width files by setting lengths on the fields in the fields tab.

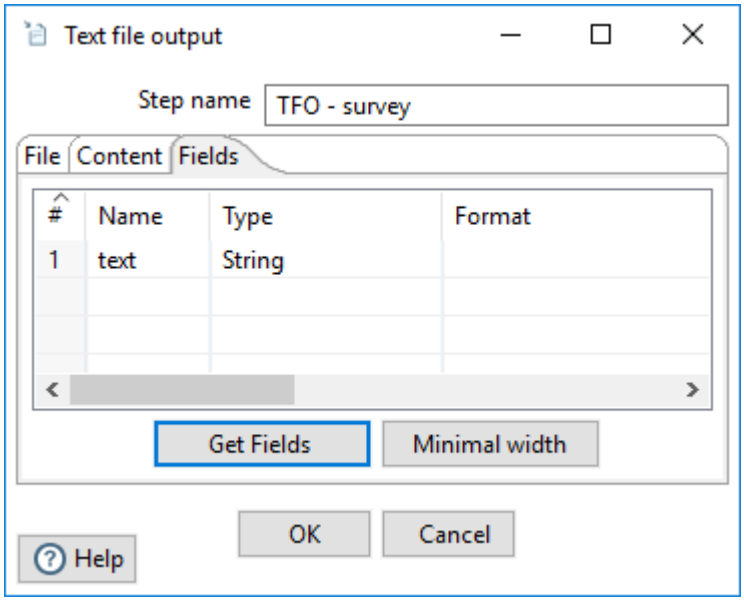
> It is not possible to execute this step in parallel to write to the same file. In this case, you need to set the option "Include stepnr in filename" and later merge the files.

1. Drag the 'Text File Output' step onto the canvas.
2. Create a hop from the 'Append body' step.
3. Double-click on the step, and configure the following properties:

   File: `C:\Pentaho Training\DI 1000\Module 3 - Datasources\Lesson 1 - Working with Files\Demo 3-1-2 - Write Text File\survey`



4. Click on the Field tab, and click on the 'Get Fields' button.

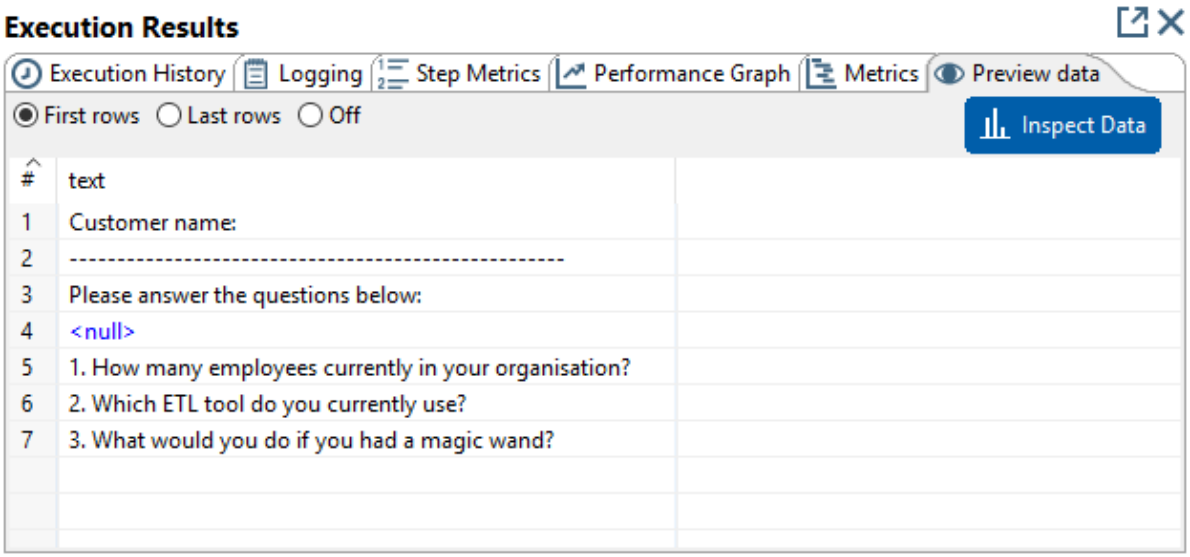


5. Close step.

- The 'text' stream field is written as a survey.txt file.

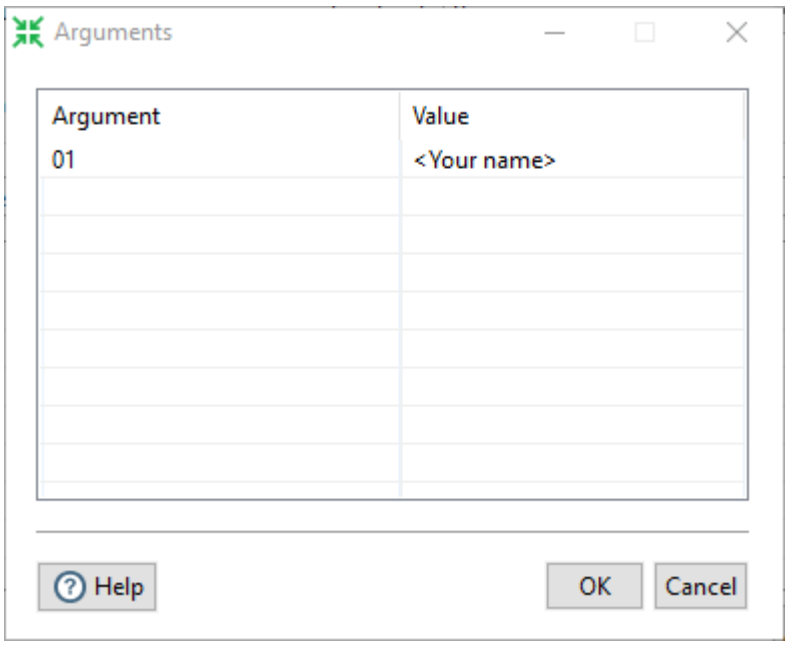
RUN the Transformation

1. Click the Run button in the Canvas Toolbar.



To enter the command line arguments for the Customer Name:

2. Click on the Arguments (Legacy) button and enter the following:



3. View the survey.txt file.

```
Customer name: James O'Reilly
---------------------------------------------------
Please answer the questions below:

1. How many employees currently in your organisation?
2. Which ETL tool do you currently use?
3. What would you do if you had a magic wand?
```

This demonstration illustrates the 'golden rules' when appending / merging data streams:

- They both must have the same layout, i.e. fieldnames in the same order in the data stream, and with the same data type.

Guided Demo 3-1-3: Excel Writer

Introduction

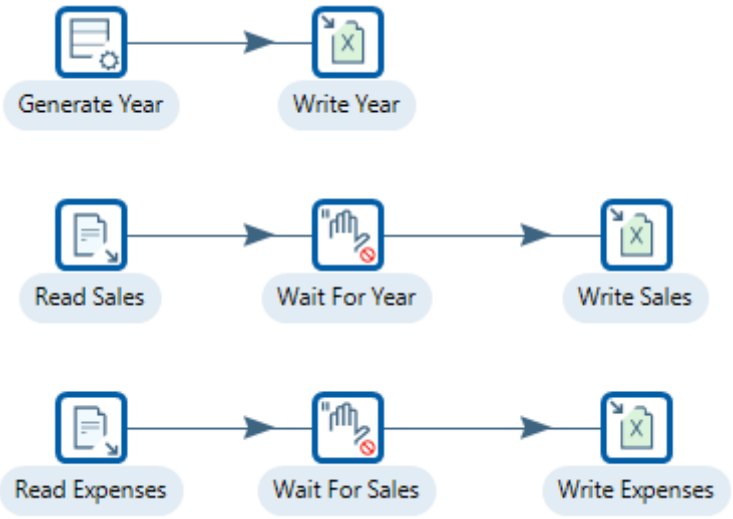
Steel Wheels wish to automate their Half Yearly Sales and Expenses Report in Excel. The ETL process has been broken down into various workflows, resulting in writing data to an Excel template, once previous workflows have been completed.

Objectives

In this guided demonstration, you will:

- Configure the following steps:
  - Excel Writer
  - Block Step

Transformation



To create a new transformation:

1. In Spoon, click File > New > Transformation.

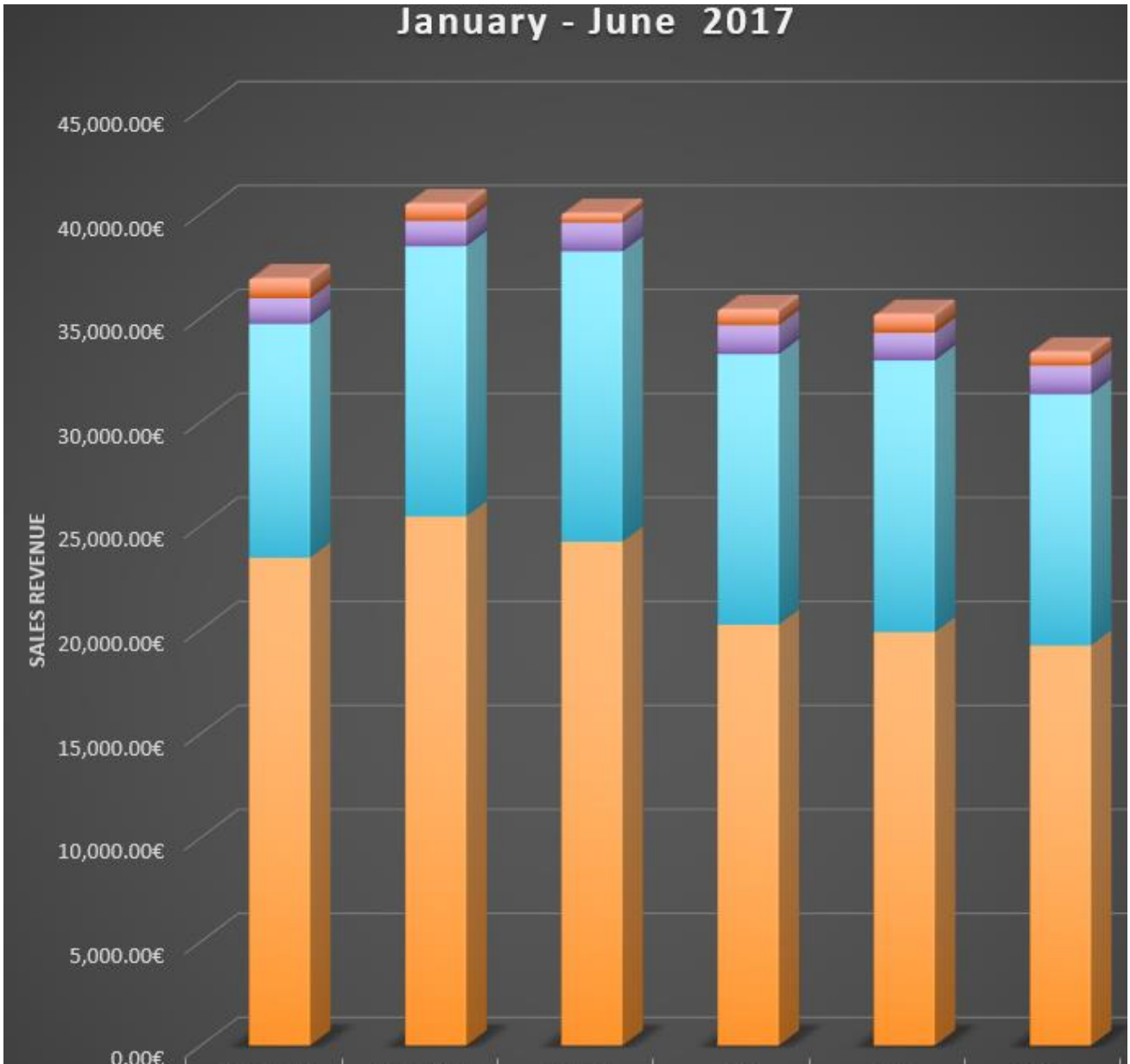
Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

- By clicking New, then Transformation
- By using the CTRL-N hot key

Excel Template

The various stages of the transformation write data to a template.xlsx. The template has 3 worksheets:

- Sales Chart - this worksheet creates a 3D stacked graph based on the data in the SourceData worksheet.



	JANUARY	FEBRUARY	MARCH	APRIL	MAY	JUNE
Planes	956.12€	834.56€	457.76€	765.32€	898.11€	667.49€
Trains	1,231.29€	1,227.98€	1,395.33€	1,399.90€	1,335.90€	1,376.98€
Motorcycles	11,244.21€	12,987.69€	13,954.09€	13,006.33€	13,065.31€	12,087.74€
Classic Cars	23,455.22€	25,442.11€	24,222.89€	20,233.11€	19,876.22€	19,233.56€

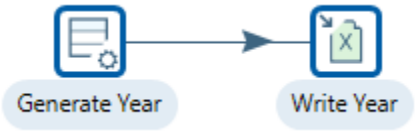
- SourceData - the datasheet. This is the main output from the various stages of the Transformation.

	A	B	C	D	E	F	G
1	**Year**	**2017**					
2							
3		**JANUARY**	**FEBRUARY**	**MARCH**	**APRIL**	**MAY**	**JUNE**
4	**PRODUCTLINE**						
5	**Classic Cars**	23,455.22€	25,442.11€	24,222.89€	20,233.11€	19,876.22€	19,233.56€
6	**Motorcycles**	11,244.21€	12,987.69€	13,954.09€	13,006.33€	13,065.31€	12,087.74€
7	**Trains**	1,231.29€	1,227.98€	1,395.33€	1,399.90€	1,335.90€	1,376.98€
8	**Planes**	956.12€	834.56€	457.76€	765.32€	898.11€	667.49€
9							
10	**Total Sales**	35,655.55€	39,264.36€	38,634.74€	34,004.76€	33,839.64€	31,988.79€
11							
12	**EXPENSES**						
13	**Advertising**	2,055.22€	2,542.11€	2,422.89€	2,033.11€	1,986.22€	1,933.56€
14	**Cost of Goods**	100.32€	103.23€	140.23€	130.23€	120.33€	121.34€
15	**Salary**	11,020.80€	11,020.80€	11,020.80€	9,350.10€	9,350.10€	12,350.60€
16	**Lease**	223.23€	223.23€	223.23€	223.23€	223.23€	223.23€
17	**Miscellaneous**	10.30€	0.00€	209.99€	3.99€	0.00€	12.23€
18	**Overhead**	90.23€	90.23€	78.90€	90.23€	78.90€	0.00€
19	**Total Expenses**	13,500.10€	13,979.60€	14,096.04€	11,830.89€	11,758.78€	14,640.96€
20							
21	**PROFIT**	22,155.45€	25,284.76€	24,538.70€	22,173.87€	22,080.86€	17,347.83€

- The Year has been formatted using Excel

Workflow 1 - Write Year

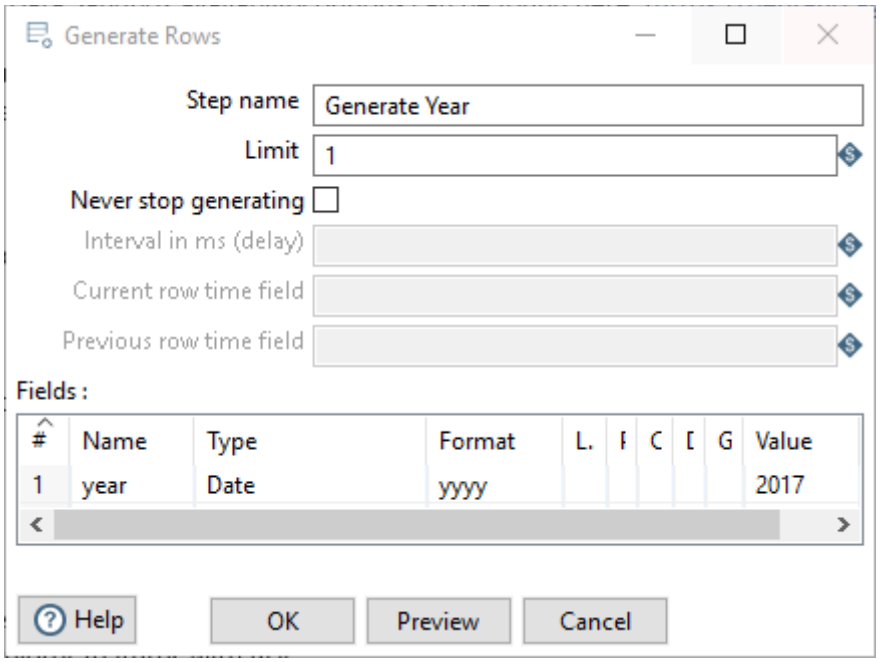
The first workflow is to write the current Year to the SourceData worksheet in the template.xlsx



Generate Rows - Year

Generate rows outputs a specified number of rows. By default, the rows are empty; however, they can contain several static fields. This step is used primarily for testing purposes. It may be useful for generating a fixed number of rows, for example, you want exactly 12 rows for 12 months. Sometimes you may use Generate Rows to generate one row that is an initiating point for your transformation.

1. Drag the 'Generate Rows' step onto the canvas.
2. Double-click on the step, and configure the following properties:



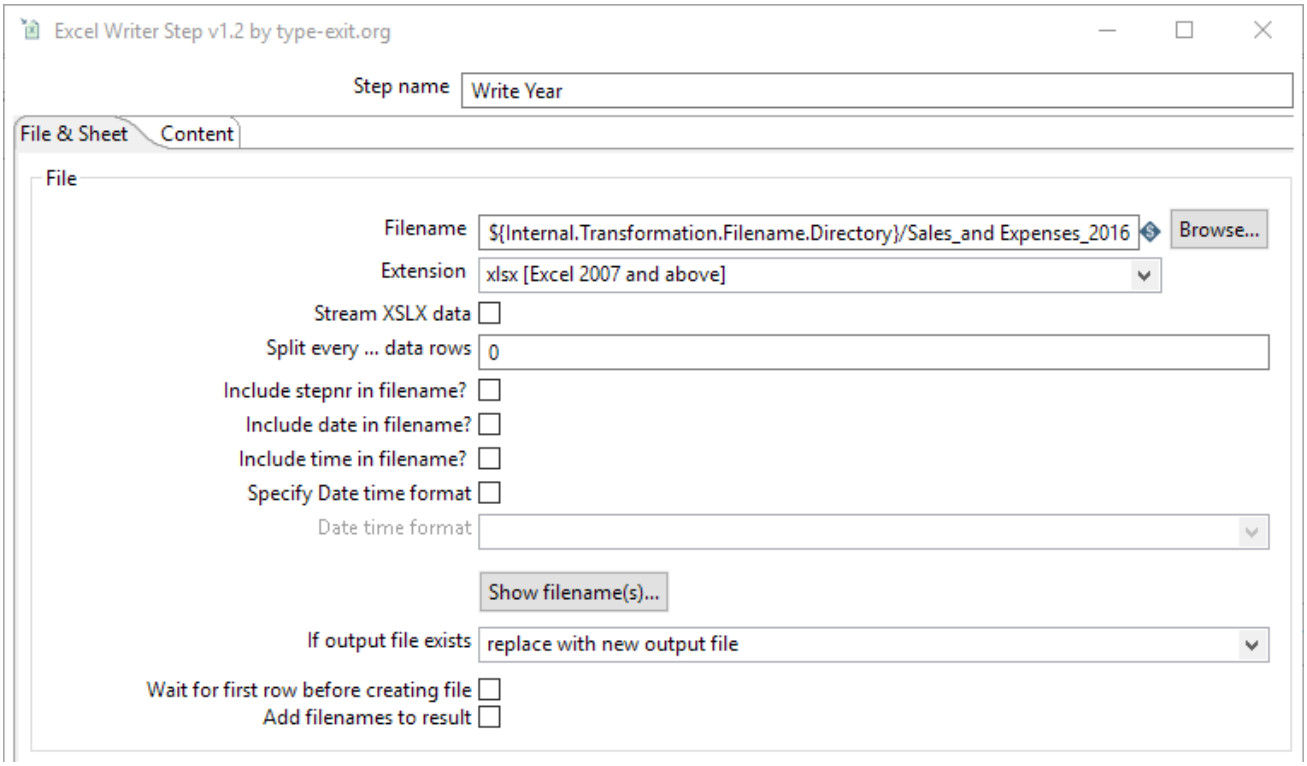
3. Close step.

- Generates a record that holds the Year value – 2017 – in the year stream field.
- The Excel template will also need to be formatted yyyy to interpret the Date.

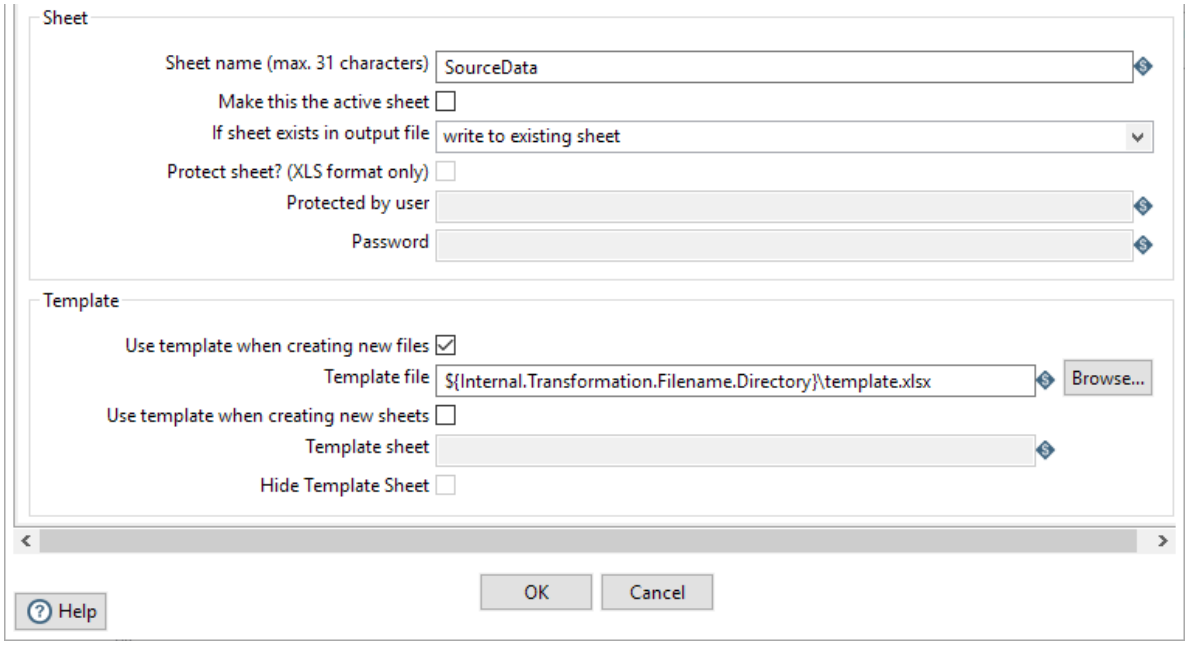
Excel Writer - Write Year

The Microsoft Excel Writer step writes incoming rows into an MS Excel file. It supports both the xls and xlsx file formats. The xlsx format is usually a good choice when working with template files, as it is more likely to preserve charts and other misc objects in the output. The proprietary (binary) xls format is not as well understood and deciphered, so moving/replicating nontrivial xls content in non-MS software environments is usually problematic.

1. Drag the 'Excel writer' step onto the canvas.
2. Create a hop from the 'Year' step.
3. Double-click on the step, and configure the following properties:



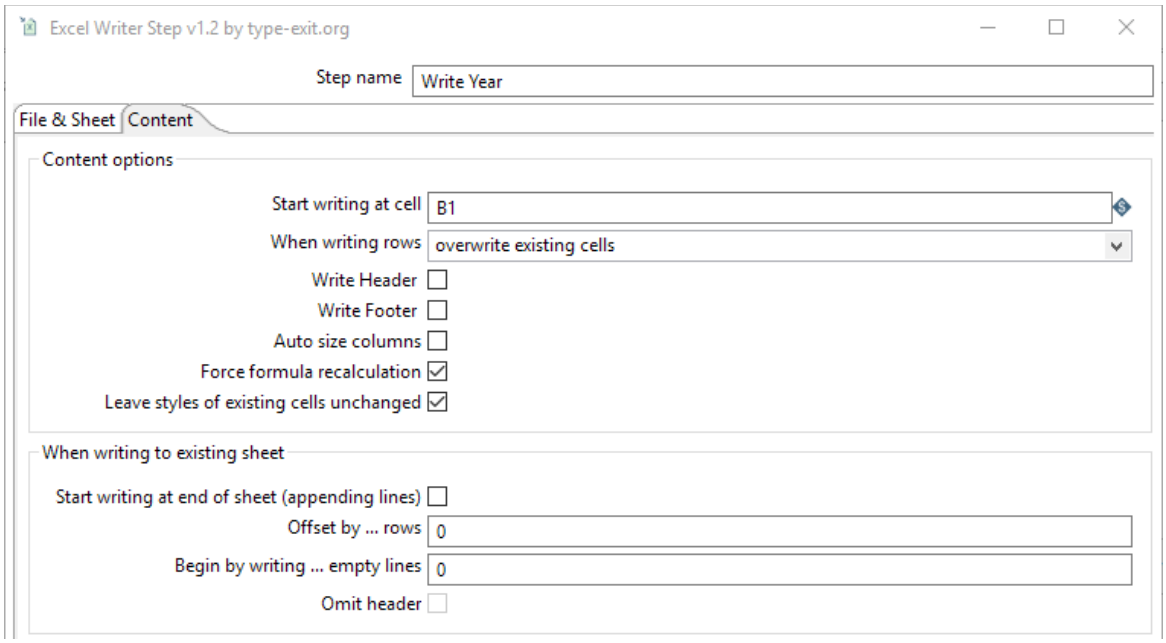
- The filename sets the name and output directory for the Excel Workbook:
  File: `C:\Pentaho Training\DI 1000\Module 3 - Datasources\Lesson 1 - Working with Files\Demo 3-1-3 - Write Excel File \Sales_and_Expenses_2017.xlsx`
- The option: replace with new output file, results in a new Excel Workbook file overwriting previous versions every time the Transformation is Run.



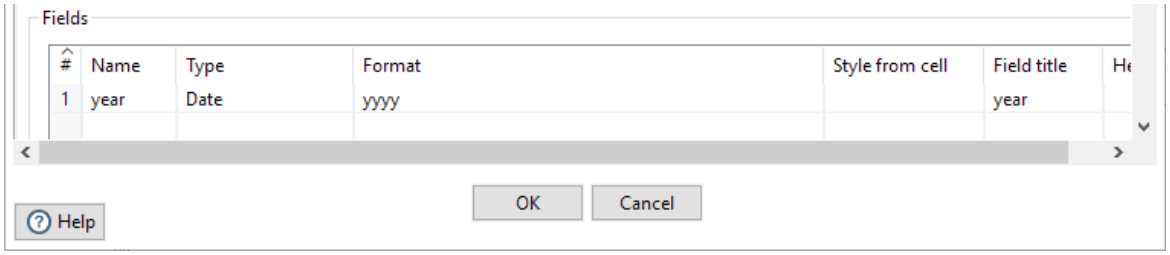
- Write the output to the SourceData Excel worksheet.
- Use template.xlsx as the template.

  File: `C:\Pentaho Training\DI 1000\Module 3 - Datasources\Lesson 1 - Working with Files\Demo 3-1-3 - Write Excel File \template.xlsx`

4. Click on the Content tab, and configure the following properties:



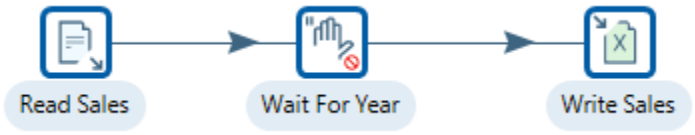
- Write the Year stream field to cell B1, on the SourceData worksheet.



5. Click on 'Get Fields' button. Click OK

Workflow 2 - Write Sales

The second workflow is to repeat the workflow for Sales, with the addition of a 'Blockstep', which ensures the 'Write Year' workflow has been completed, before writing to the template.xlsx.

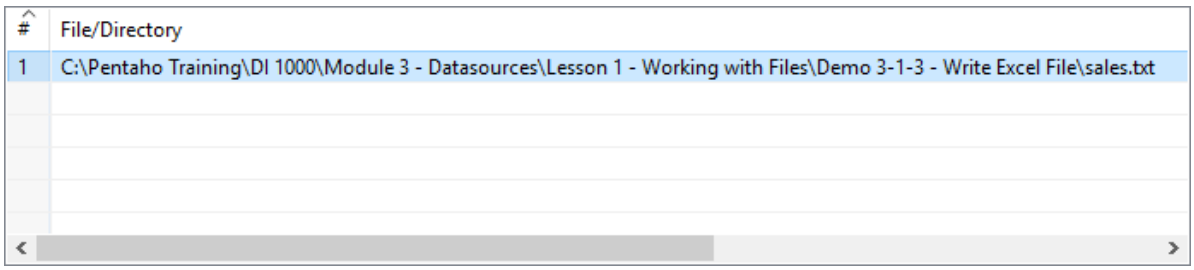


Text File Input - Read Sales

The Text File Input step is used to read data from a variety of different text-file types. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

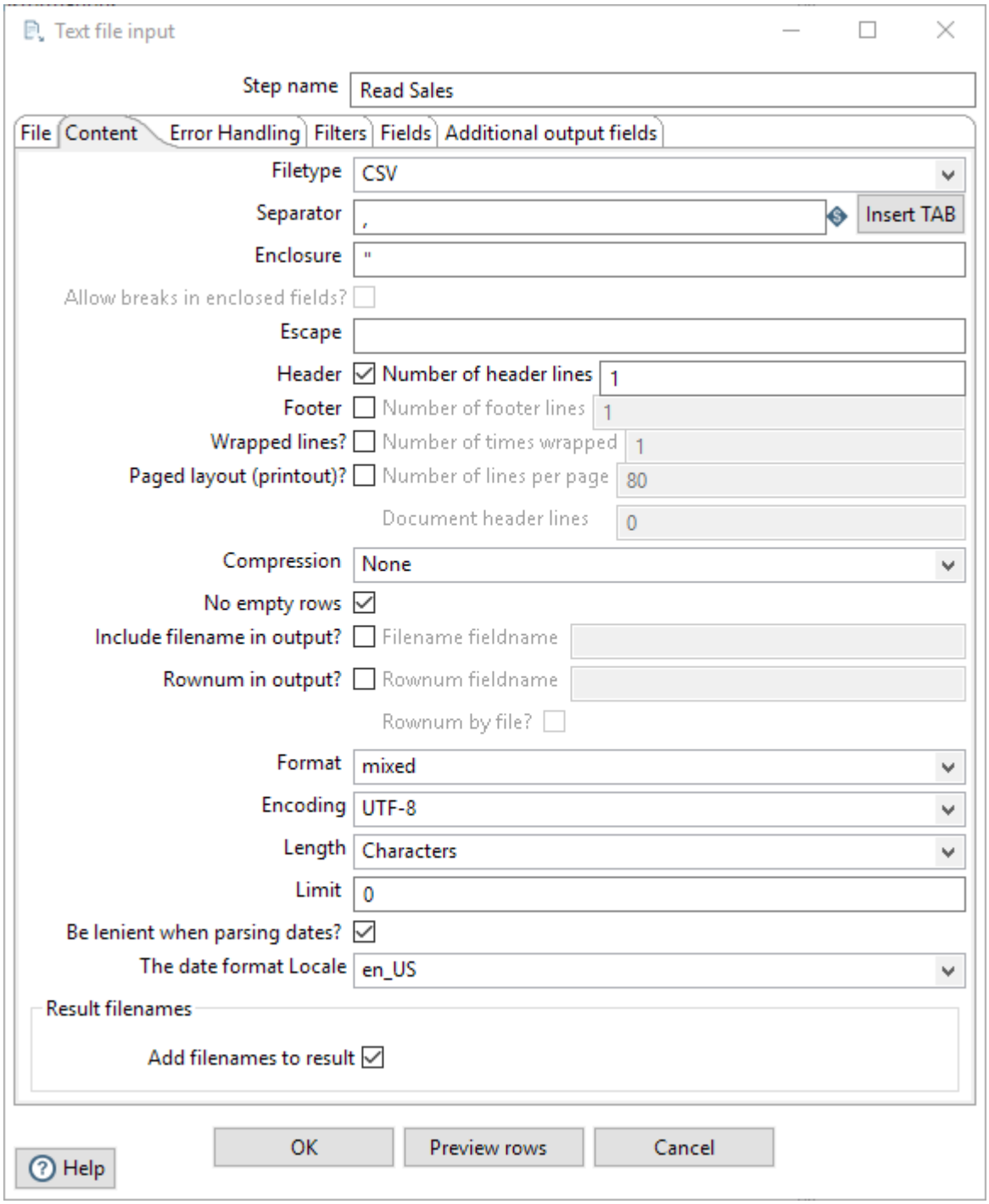
The Text File Input step provides you with the ability to specify a list of files to read, or a list of directories with wild cards in the form of regular expressions. In addition, you can accept filenames from a previous step making filename handling more even more generic.

1. Drag the 'Text File Input' step onto the canvas.
2. Double-click on the step, and configure the following properties:



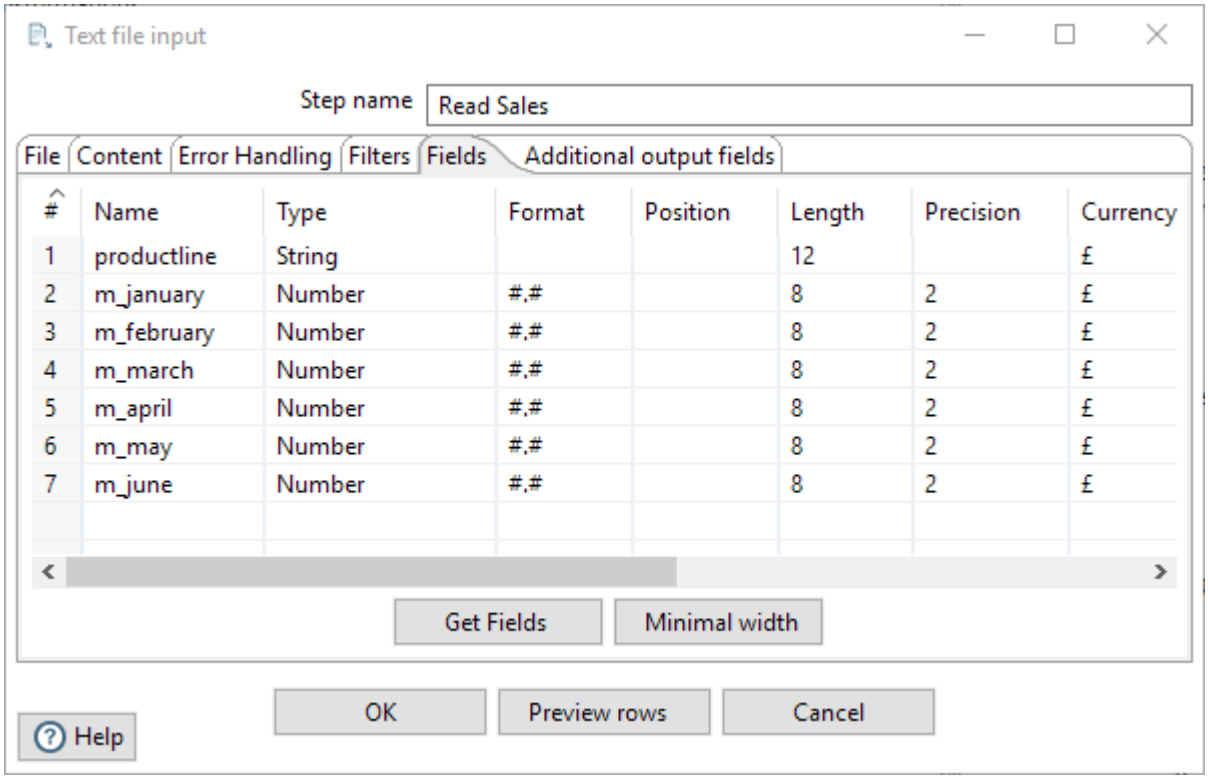
- Point to the:
  File: `C:\Pentaho Training\DI 1000\Module 3 - Datasources\Lesson 1 - Working with Files\Demo 3-1-3 - Write Excel File\sales.txt` file.

3. Click on the Content tab, and configure the following properties:



- Ensure the Header is selected.
- No empty rows
- Mixed Format

4. Click on the Fields tab, and click on 'Get Fields' button.:



- Returns the Header values as stream fields.

5. Click OK.

Block until Step Finish - Wait Year

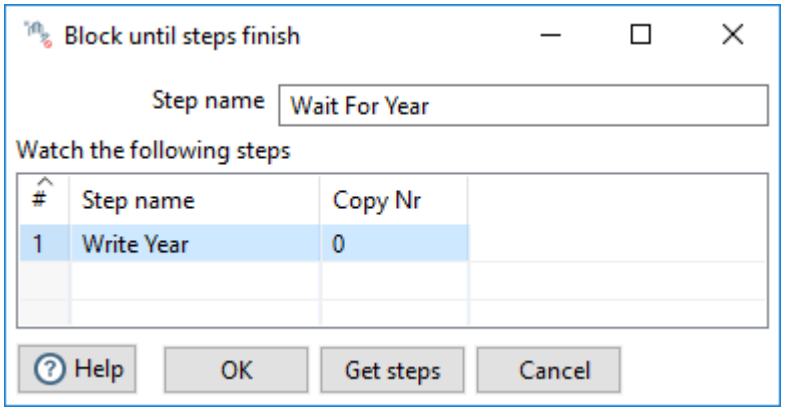
This step simply waits until all the step copies that are specified in the dialog have finished.

You can use it to avoid the natural concurrency (parallelism) that exists between transformation step copies.

Option	Description
Step name	Name of the step. *Note*: This name must be unique in a single transformation.
Watch the following steps	Use this grid to specify the steps to wait for.
Get steps	Push this button to auto-fill the "Watch the following steps" grid with all steps available in the transformation.

Column	Description
Step name	The name of the step to wait for.
CopyNr	The (0-based) copy number of the step. If the named step has an explicit setting for "Change number of copies to start", and you want to wait for all copies to finish, you'll need to enter one row in the grid for each copy, and use this column to specify which copy of the step to wait for. For the default number of copies (1), the CopyNr is always 0.

1. Drag the 'Block this step until steps finish' step onto the canvas.
2. Create a hop from the 'Read Sales' step.
3. Double-click on the step, and configure the following properties:

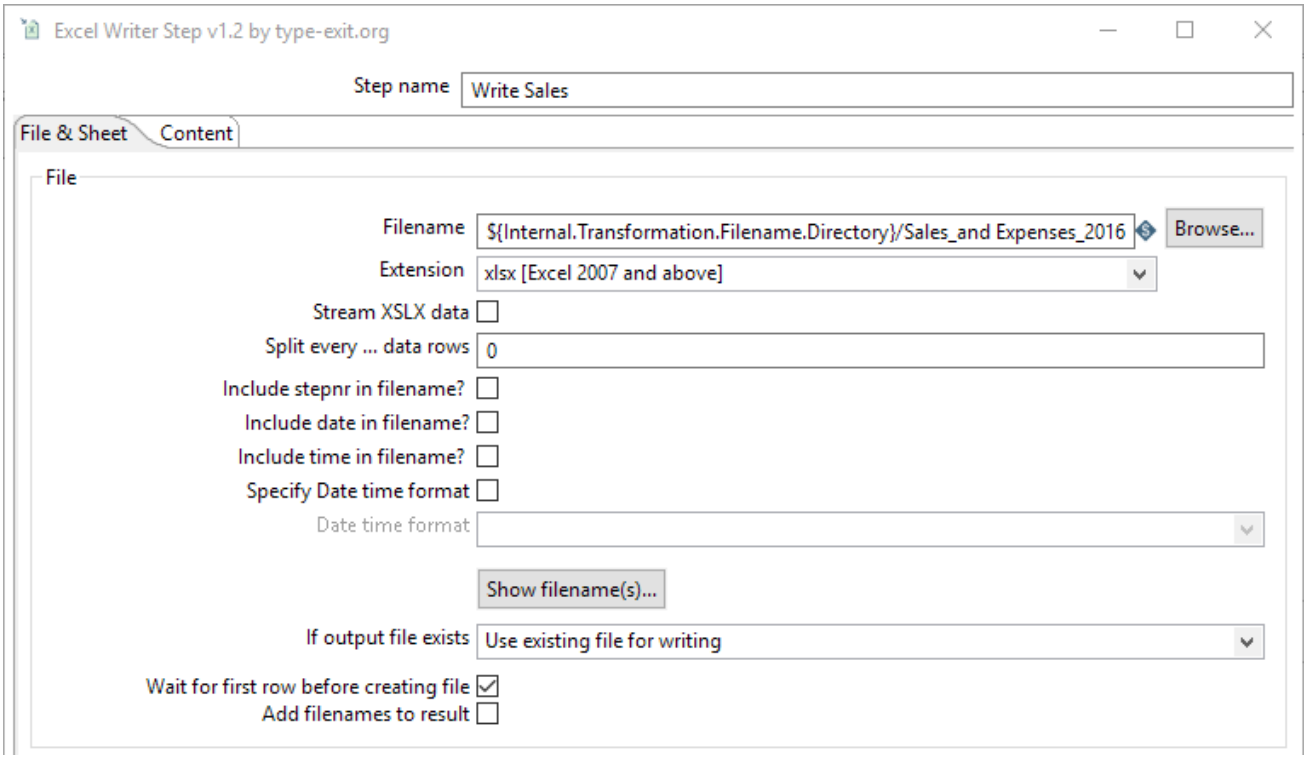


- This will result in the workflow being blocked until the Write Year step has been completed.

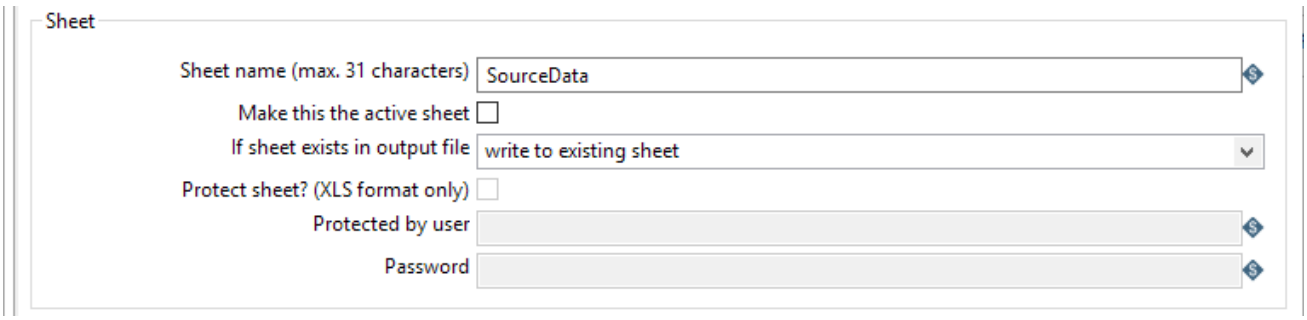
Excel Writer - Write Sales

The Microsoft Excel Writer step writes incoming rows into an MS Excel file. It supports both the xls and xlsx file formats. The xlsx format is usually a good choice when working with template files, as it is more likely to preserve charts and other misc objects in the output. The proprietary (binary) xls format is not as well understood and deciphered, so moving/replicating nontrivial xls content in non-MS software environments is usually problematic.

1. Drag the 'Excel writer' step onto the canvas.
2. Create a hop from the 'Wait Year' step.
3. Double-click on the step, and configure the following properties:

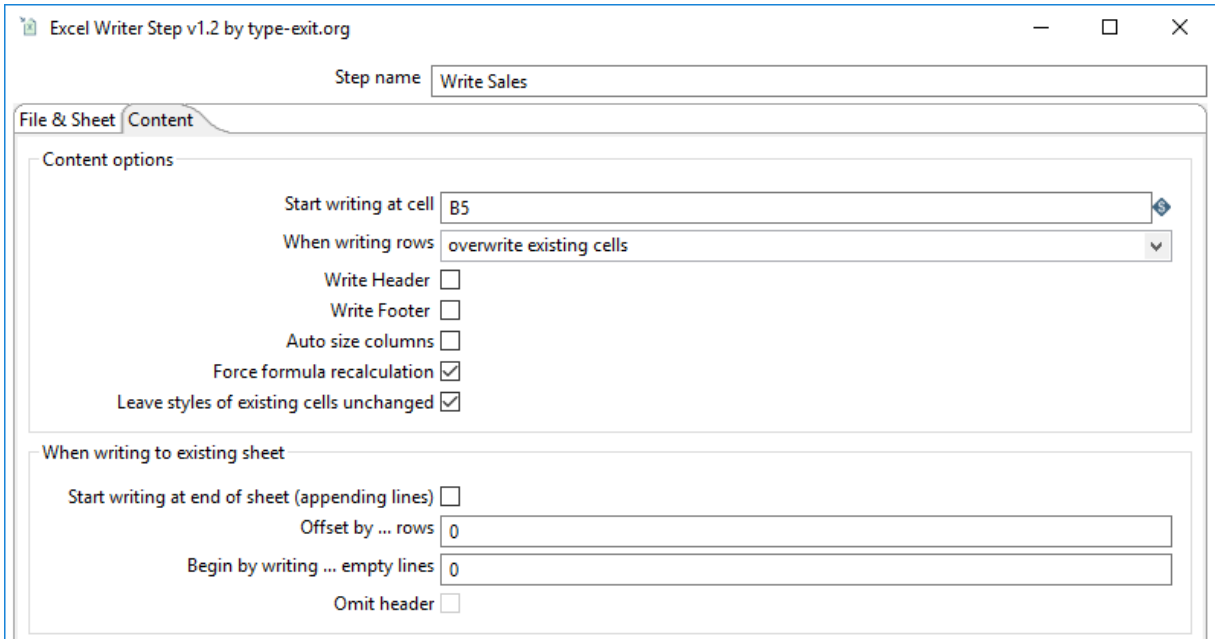


- The filename sets the name and output directory for the Excel Workbook:
  File: `C:\Pentaho Training\DI 1000\Module 3 - Datasources\Lesson 1 - Working with Files\Demo 3-1-3 - Write Excel File \Sales_and_Expenses_2017.xlsx`
- The option: Use existing file for writing, writes the data to the existing Sales_and_Expenses_2017.xlsx file

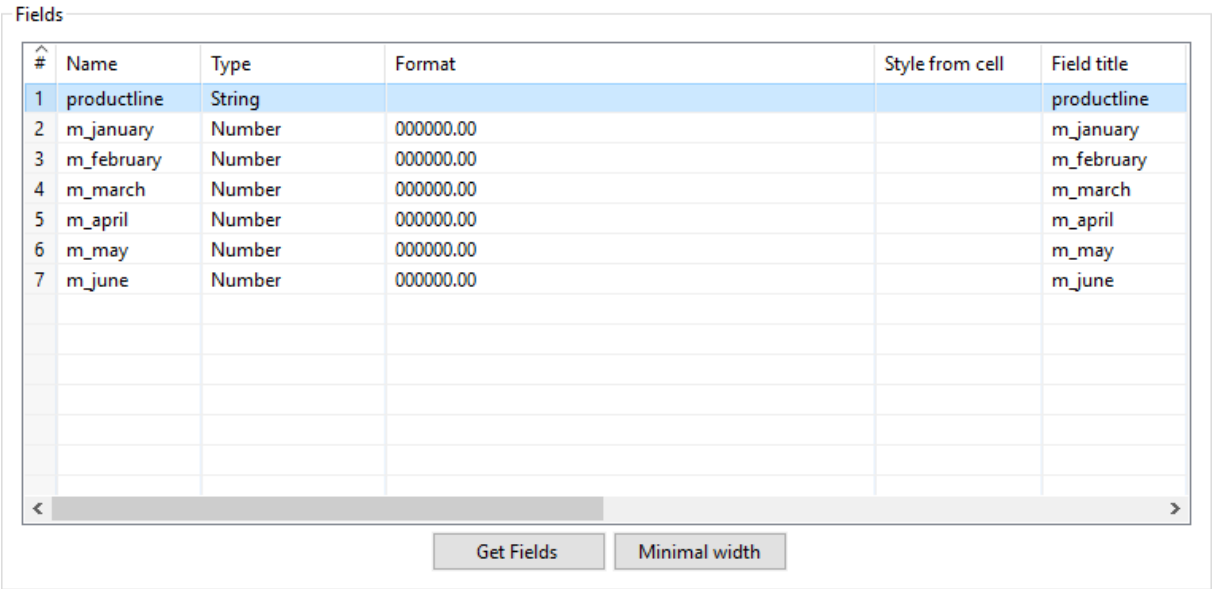


- Write the output to the SourceData Excel worksheet.
- No need for the template as the data is now being written to the file.

4. Click on the Content tab, and configure the following properties:



- Write the m_MONTH stream fields starting at cell B5, on the SourceData worksheet.



5. Click on the 'Get Fields' button.
6. Delete the productline field, as its not required. The template already has the fieldname and you are just writing the data, starting at cell B5.
7. Click OK.

Workflow 3 – Write Expenses

The third workflow is similar to workflow 2, however, for Expenses.

Try configuring the steps without a guide, based on the following configuration properties:

Read Expenses

- Uses the expenses.txt file

Wait for Sales

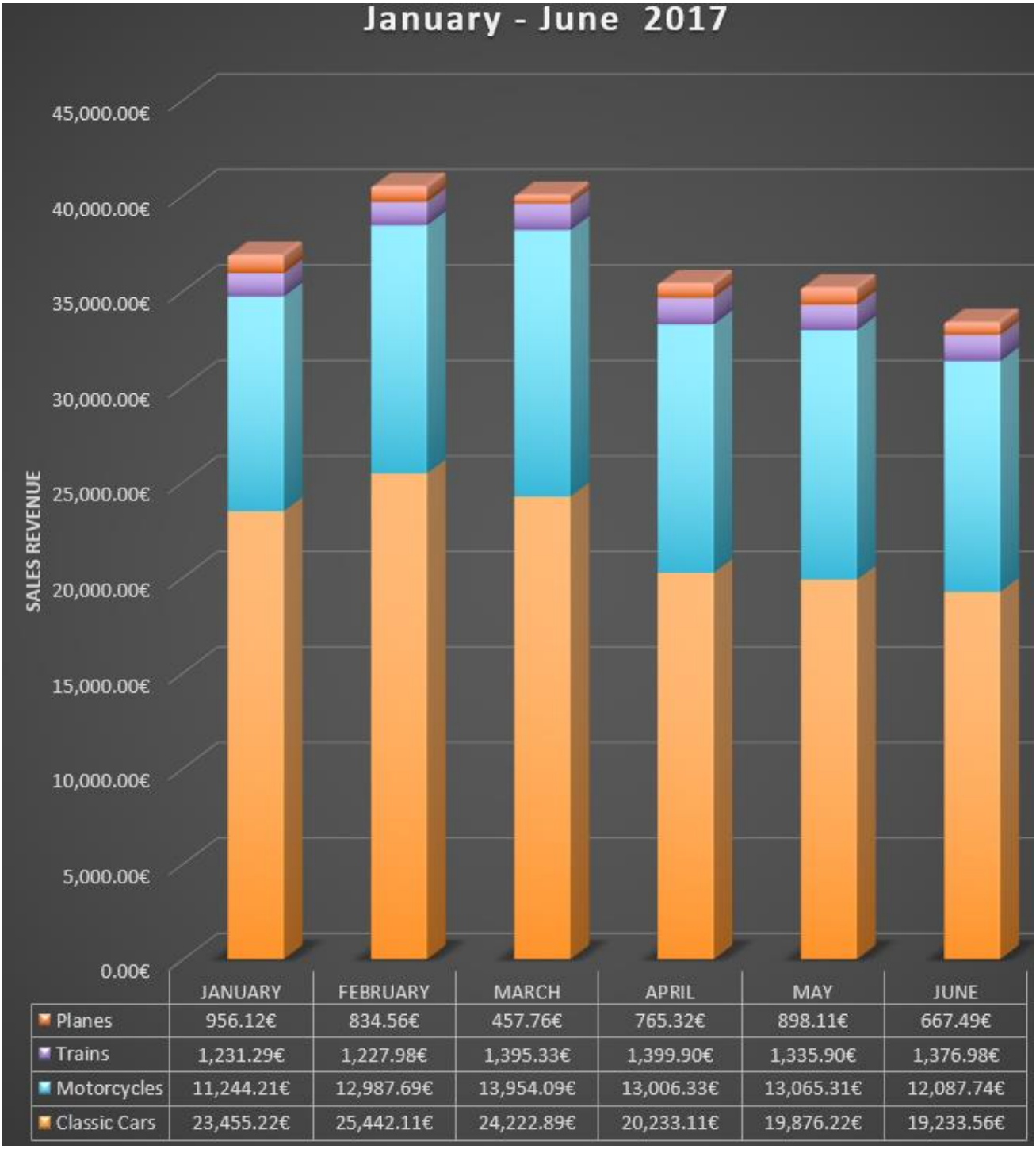
- Wait for 'Write Sales' step

Write Expenses

- Start writing at cell B13
- Remember to remove expense_type field.

RUN the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Open the Sales_and_Expenses_2017.xlsx file.



	JANUARY	FEBRUARY	MARCH	APRIL	MAY	JUNE
Planes	956.12€	834.56€	457.76€	765.32€	898.11€	667.49€
Trains	1,231.29€	1,227.98€	1,395.33€	1,399.90€	1,335.90€	1,376.98€
Motorcycles	11,244.21€	12,987.69€	13,954.09€	13,006.33€	13,065.31€	12,087.74€
Classic Cars	23,455.22€	25,442.11€	24,222.89€	20,233.11€	19,876.22€	19,233.56€

This demonstration illustrates the concept of parallelism. As each Transformation step is initialized in parallel, i.e. their own thread, then blocking steps must be added to the workflow, to prevent data being simultaneously written to the Excel SourceData worksheet.

Guided Demo 3-1-4: Read XML

Introduction

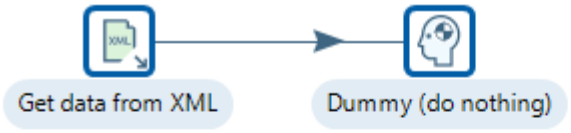
Steel Wheels has some data sources in XML format. This guided demonstration illustrates the 3 data source options for retrieving XML data.

Objectives

In this guided demonstration, you will:

- Configure the following step:
  - Get data from XML
    - File
    - URL
    - Stream

Transformation



To create a new transformation:

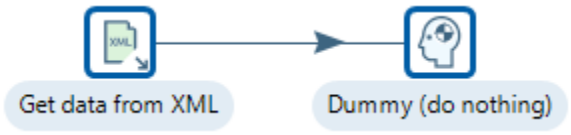
1. In Spoon, click File > New > Transformation.

Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

- By clicking New, then Transformation
- By using the CTRL-N hot key

Workflow 1 – Get data from XML (File)

In this workflow, an XML file is parsed via an X-path to retrieve the required dataset.



Get data from XML

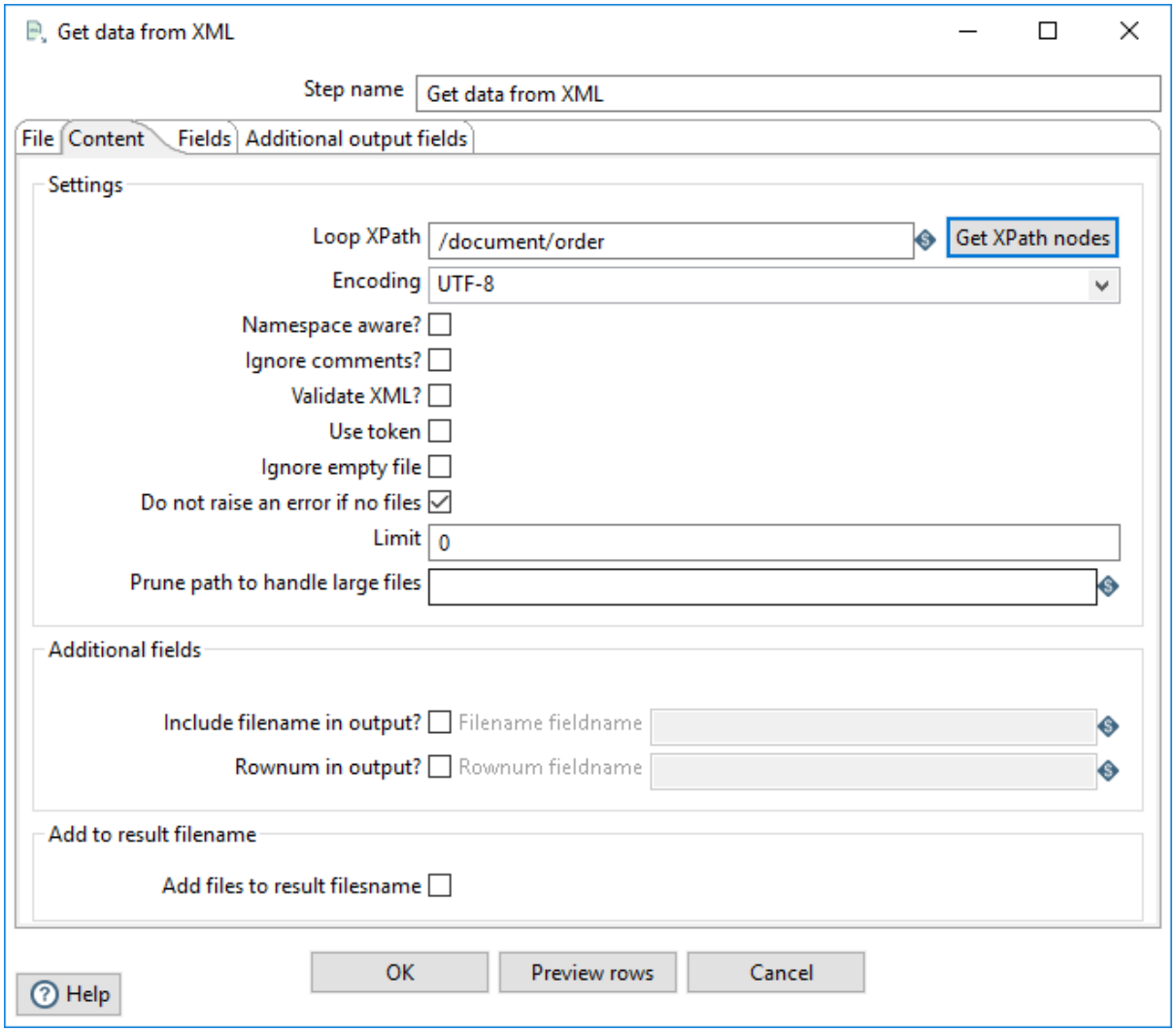
This step provides the ability to read data from any type of XML file using XPath specifications.

> HTTP URL requests: You should be aware that this component will re-query the URL for each lookup request. Therefore, if you are using this component to query a HTTP URL for an XML document, you should consider using the "Loop XPath" and "Prune path" parameters - these will help you avoid re-querying for the XML document.
>
> This component seems to make a HTTP HEAD request prior to the actual HTTP GET - if your XML document is a dynamic query (ASP/PHP/etc.), you should be aware that this HTTP HEAD request may result in an additional call to your URL.
>
> Please see also the XML Input Stream (StAX) step that uses a completely different approach to solve use cases with very big and complex data structures and the need for very fast data loads.

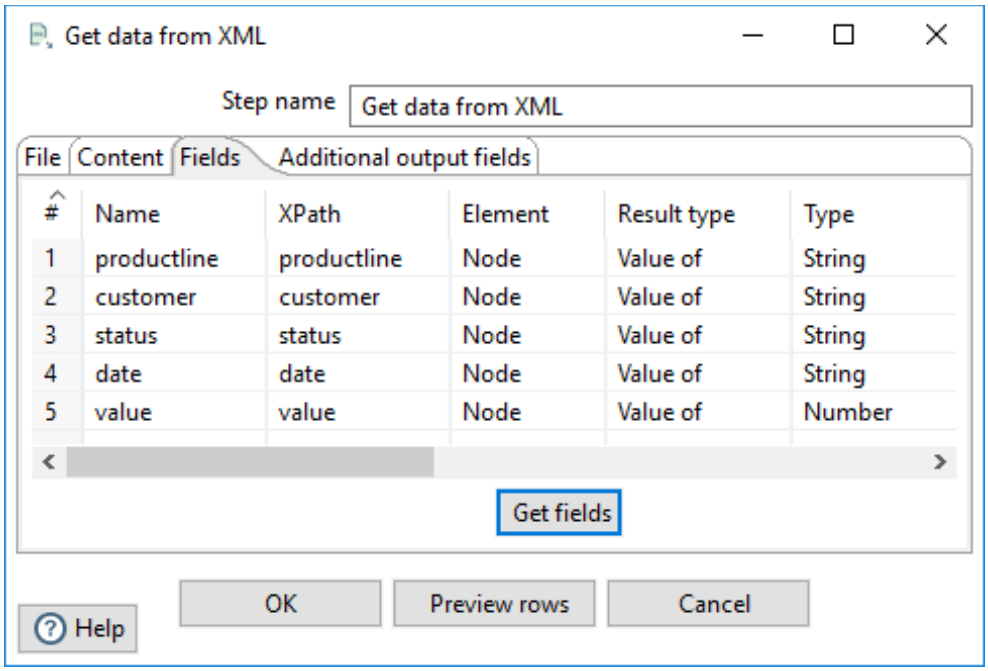
1. Drag the 'Get data from XML' step onto the canvas.
2. Double-click on the step, and configure the following properties:

#	File/Directory
1	C:\Pentaho Training\DI 1000\Module 3 - Datasources\Lesson 1 - Working with Files\Demo 3-1-4 - Read XML\orders.xml

3. Click on the Content tab, and configure the following properties:



4. Click on the Fields tab, and then on the 'Get Fields' button.



5. Click OK.

Dummy

The Dummy step does not do anything. Its primary function is to be a placeholder for testing purposes. For example, to have a transformation, you need at least two steps connected to each other.

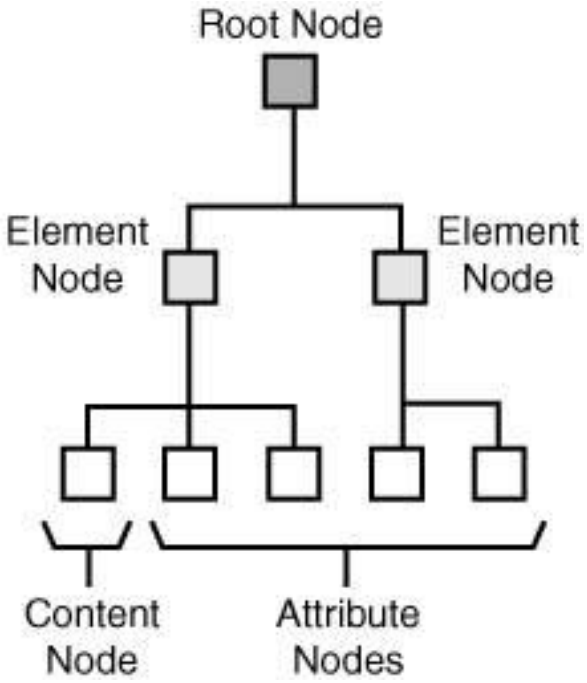
1. Drag a 'Dummy' step onto the canvas.
2. Create a hop from the 'Get data from XML' step.
3. Close the step.

XPath

XPath is a set of rules used for getting information from an XML document. In XPath, XML documents are treated as trees of **nodes**. There are several types of nodes; elements, attributes, and texts are some of them. As an example, document, and order are some of the nodes in the sample file.

Among the nodes there are relationships. A node has a parent, zero or more children, siblings, ancestors, and descendants depending on where the other nodes are in the hierarchy.

To select a node in an XML document, you should use a path expression relative to a **current node**.



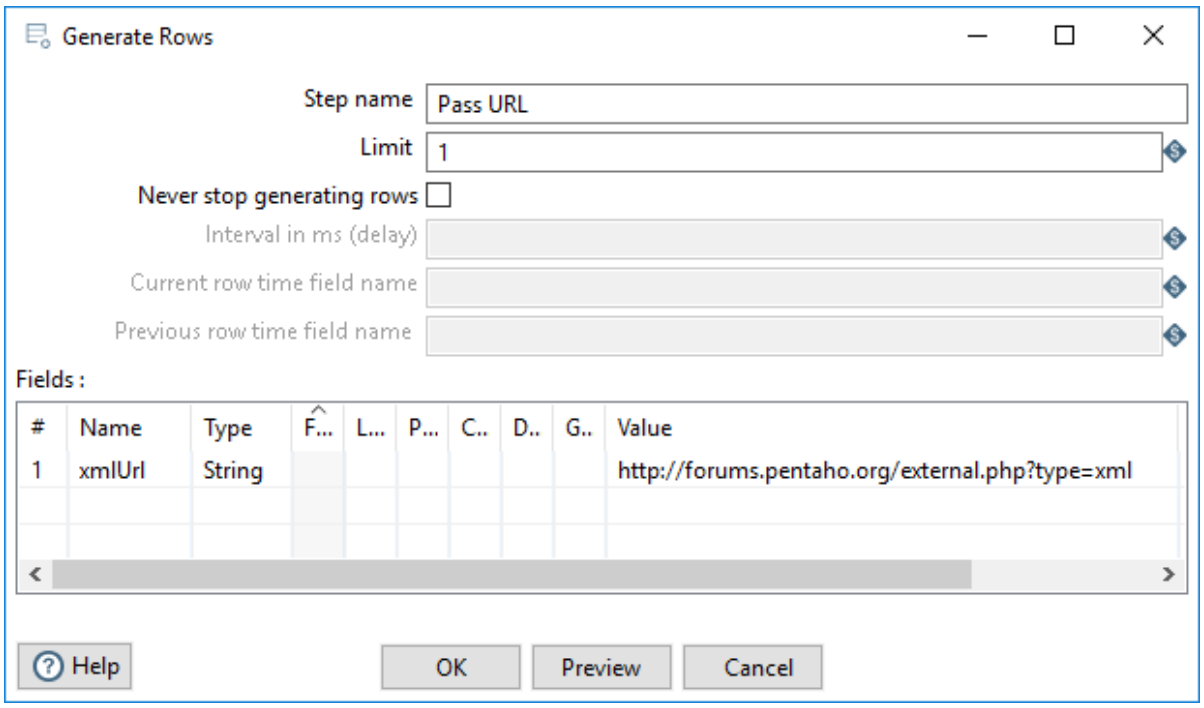
Workflow 2 – Get data from XML (File)

In this workflow, a URL to an XML data source is parsed via an X-path to retrieve the required dataset.

Generate Rows - Pass URL

Generate rows outputs a specified number of rows. By default, the rows are empty; however, they can contain several static fields. This step is used primarily for testing purposes. It may be useful for generating a fixed number of rows, for example, you want exactly 12 rows for 12 months.

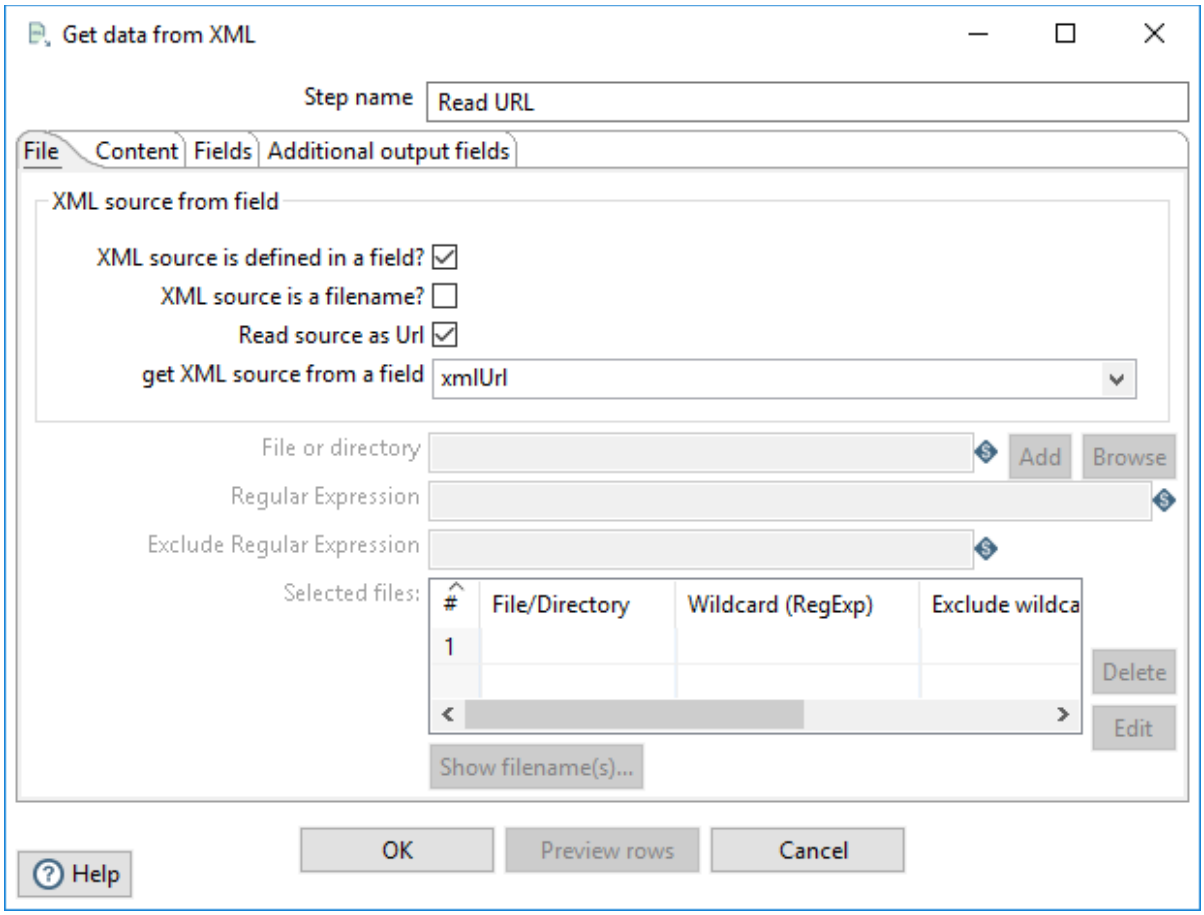
1. Drag the 'Generate Rows' step onto the canvas.
2. Double-click on the step, and configure the following properties:



3. Close the step.

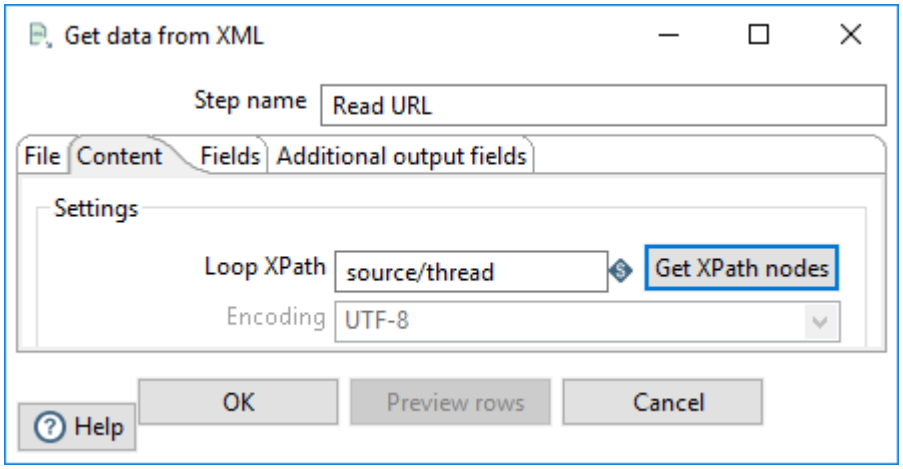
Get data from XML – Read URL

1. Drag the 'Generate Rows' step onto the canvas.
2. Create a hop from the 'Pass URL' step.
3. Double-click on the step, and configure the following properties:

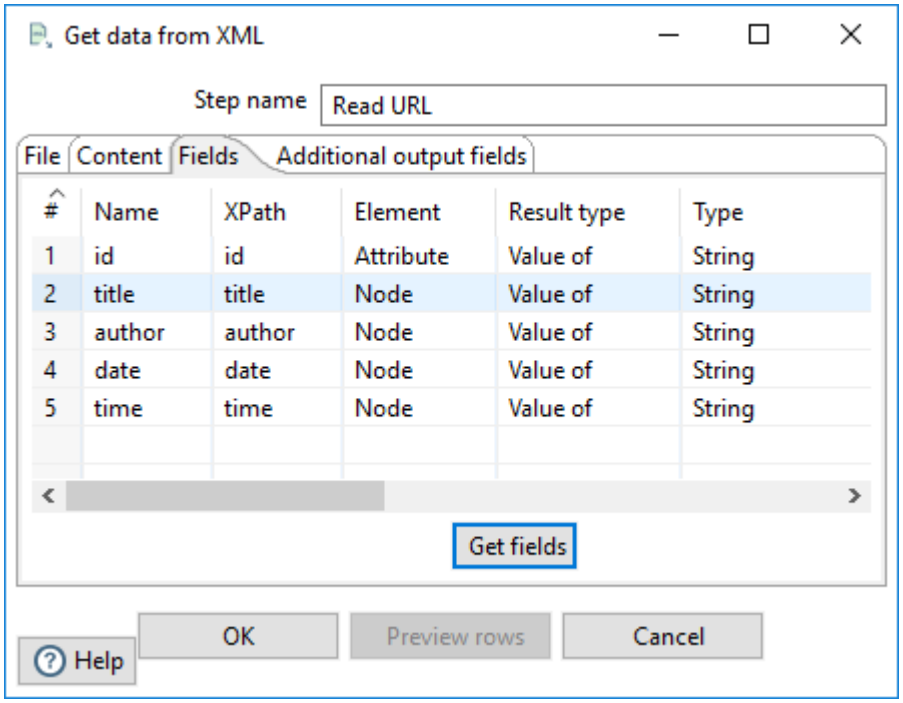


- The dataset is being parsed from a stream field xmlUrl that's being passed on from the 'Pass URL' step.

4. Click on the 'Content' tab and configure the following properties:
5. You will need to click on the 'Get XPath nodes' button and enter the URL to retrieve the XPath.



6. Click on the 'Fields' tab and configure the following properties:
7. Click on the 'Get Fields' button.



8. Close the step.

Dummy

The Dummy step does not do anything. Its primary function is to be a placeholder for testing purposes. For example, to have a transformation, you need at least two steps connected to each other.

1. Drag a 'Dummy' step onto the canvas.
2. Create a hop from the 'Read URL' step.
3. Close the step.

RUN the Transformation

1. Click the Run button in the Canvas Toolbar
2. Click on the Dummy step and Preview data.

Workflow 1

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

First rows | Last rows | Off — Inspect Data

#	productline	customer	status	date	value
1	Classic Cars	Christine Loomis	Delivered	January 2004	21.99
2	Classic Cars	Mary L. Peachin	Delivered	November 2008	24.99
3	Trains	Bob Italia	Delivered	July 1994	14.99
4	Planes	Scott M. Ascher	Delivered	March 2014	27.99
5	Motorcycles	Monty Halls	Returned	April 2007	29.99
6	Trains	Paul McCallum	Returned	June 2017	34.99
7	Boats	Jill Robinson	Delivered	November 2014	19.99

Workflow 2

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

First rows | Last rows | Off — Inspect Data

#	xmlUrl	id	title
1	http://forums.pentaho.org/external.php?type=xml	214286	Can't enable Checkpointing in pentaho DI 6.1
2	http://forums.pentaho.org/external.php?type=xml	214268	What's scope of javascript variable?
3	http://forums.pentaho.org/external.php?type=xml	214265	User Defined Java Expression - not a boolean expression
4	http://forums.pentaho.org/external.php?type=xml	214253	Schedule reports - cannot change generated content location
5	http://forums.pentaho.org/external.php?type=xml	214250	Transformation don't set variable after java filter
6	http://forums.pentaho.org/external.php?type=xml	214245	Get Variable + JOB
7	http://forums.pentaho.org/external.php?type=xml	214243	Invalid state, the Connection object is closed
8	http://forums.pentaho.org/external.php?type=xml	214238	Which pentaho bi server having dashboard creating option
9	http://forums.pentaho.org/external.php?type=xml	214237	Serialize to file for storing presorted data
10	http://forums.pentaho.org/external.php?type=xml	214235	executing batch files
11	http://forums.pentaho.org/external.php?type=xml	214232	Download CSV URL
12	http://forums.pentaho.org/external.php?type=xml	214231	Reading SPSS(.sav) file from pentaho Kettle

Introduction	Steel Wheels have several JSON data sources. In this guided demonstration, you will create a simple workflow to extract the required reporting dataset.
Objectives	In this guided demonstration, you will: • Configure the following step: ○ JSON Input
Transformation	

To create a new transformation:

1. In Spoon, click File > New > Transformation.

Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

- By clicking New, then Transformation
- By using the CTRL-N hot key

JSON is short for JavaScript Object Notation, and is a way to store information in an organized, easy-to-access manner. In a nutshell, it gives us a human-readable collection of data that we can access in a logical manner.

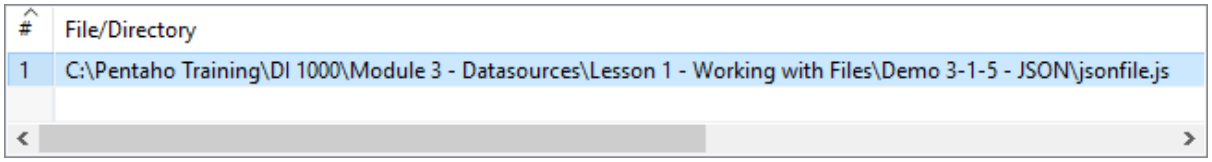
2. View the jsonfile.js

```
{ "document": {
    "order": [
      { "productline": "Classic Cars",
        "customer": "Christine Loomis",
        "status": "Delivered",
          "date": "January 2004",
        "value": 21.99
      },
      { "productline": "Classic Cars",
        "customer": "Mary L. Peachin",
        "status": "Delivered",
          "date": "November 2008",
        "value": 24.99
```

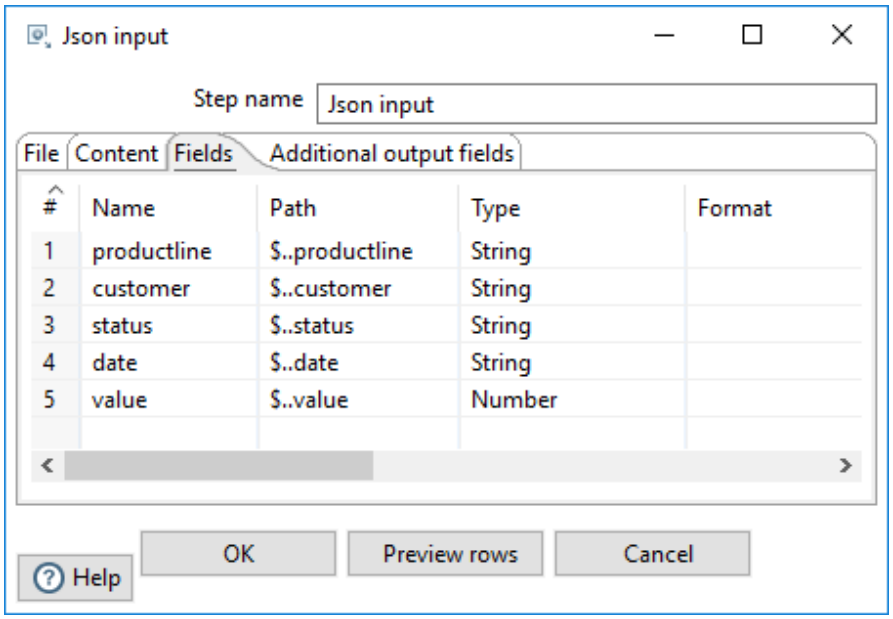
JSON Input

The JSON Input step extracts relevant portions out of JSON structures, files or incoming fields, and outputs rows.

1. Drag the 'Json Input' step onto the canvas.
2. Double-click on the step, and configure the following properties:



3. Click on the 'Fields' tab and configure the following properties:



4. Close the step.

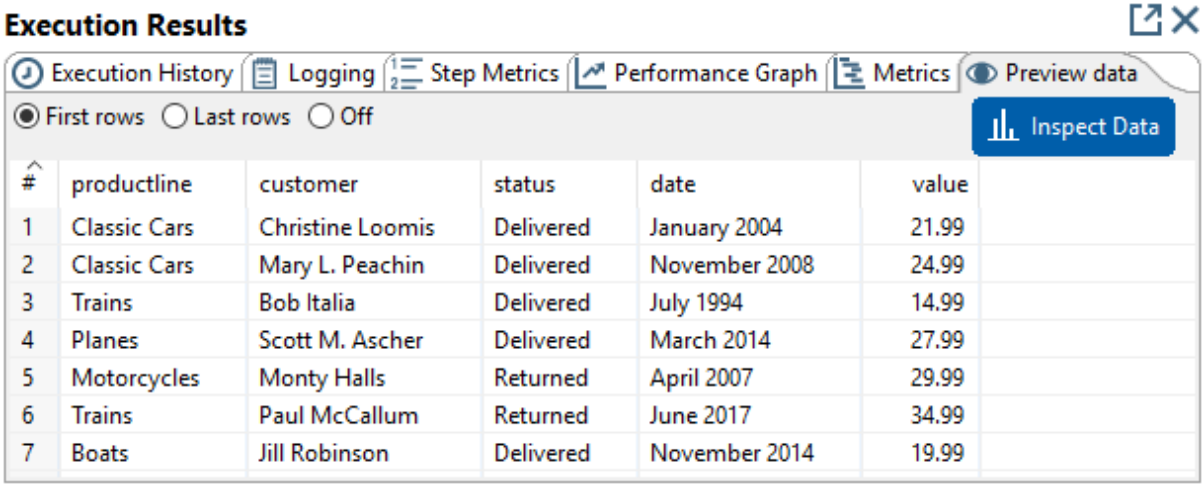
Dummy

The Dummy step does not do anything. Its primary function is to be a placeholder for testing purposes. For example, to have a transformation, you need at least two steps connected to each other.

1. Drag a 'Dummy' step onto the canvas.
2. Create a hop from the 'Json Input' step.
3. Close the step.

RUN the Transformation

1. Click the Run button in the Canvas Toolbar
2. Click on the Dummy step and Preview data.



#	productline	customer	status	date	value
1	Classic Cars	Christine Loomis	Delivered	January 2004	21.99
2	Classic Cars	Mary L. Peachin	Delivered	November 2008	24.99
3	Trains	Bob Italia	Delivered	July 1994	14.99
4	Planes	Scott M. Ascher	Delivered	March 2014	27.99
5	Motorcycles	Monty Halls	Returned	April 2007	29.99
6	Trains	Paul McCallum	Returned	June 2017	34.99
7	Boats	Jill Robinson	Delivered	November 2014	19.99

Guided Demo 3-1-6: RSS Feed (Optional)

Introduction

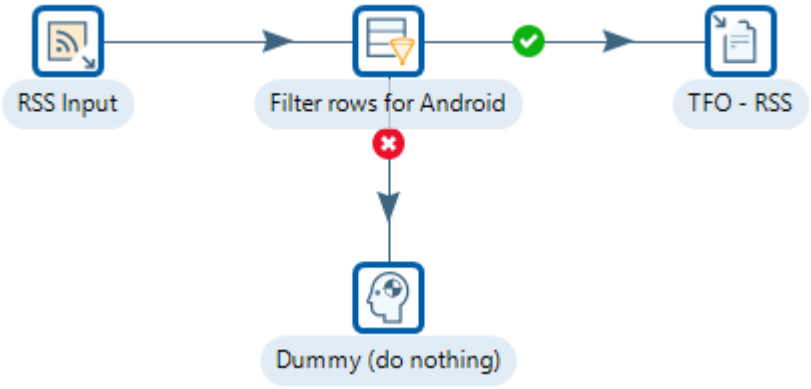
Steel Wheels have several JSON data sources. In this guided demonstration, you will create a simple workflow to extract the required reporting dataset.

Objectives

In this guided demonstration, you will:

- Configure:
  - RSS Input
  - Filter Step

Transformation



RSS (Rich Site Summary; originally RDF Site Summary; often called Really Simple Syndication) uses a family of standard web **feed** formats to publish frequently updated information: blog entries, news headlines, audio, video.

To create a new transformation:

1. In Spoon, click File > New > Transformation.

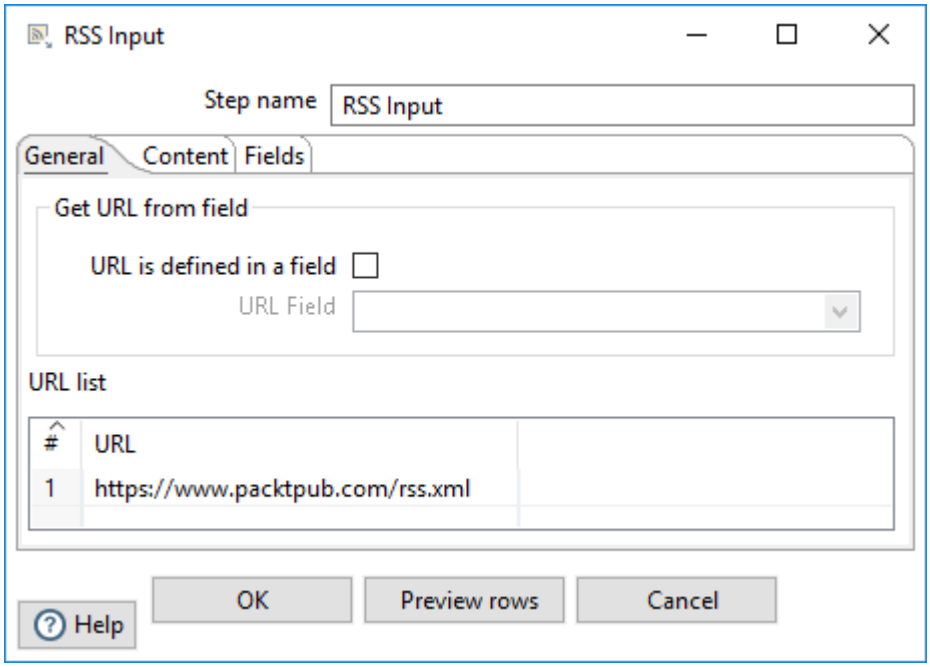
Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

- By clicking New, then Transformation
- By using the CTRL-N hot key

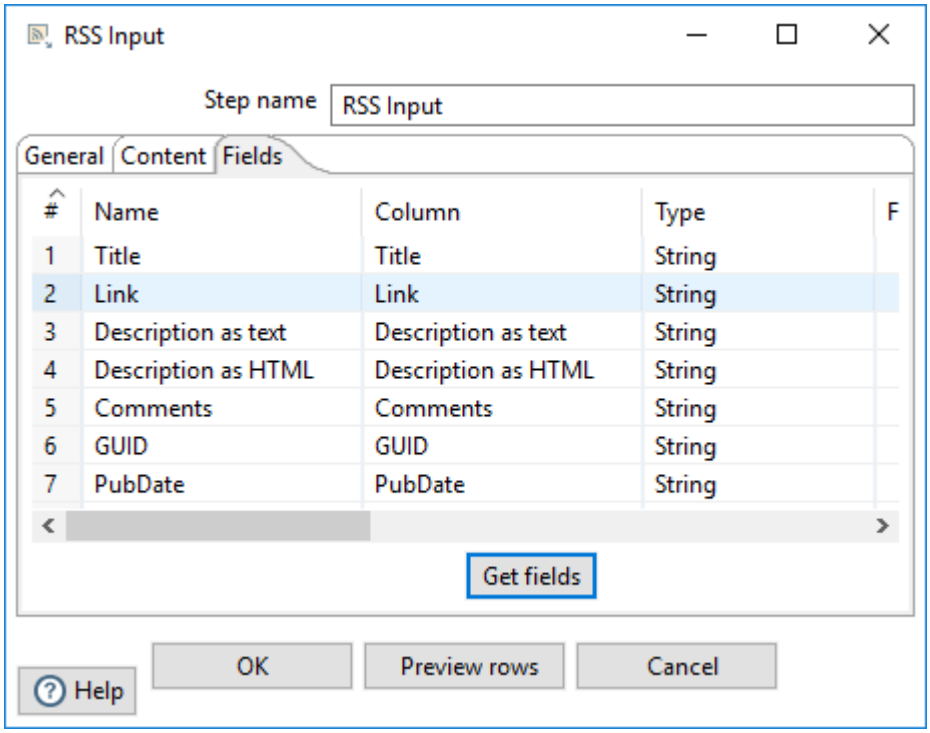
RSS Input

This step imports data from an RSS or Atom feed. RSS versions 0.91, 0.92, 1.0, 2.0, and Atom versions 0.3 and 1.0 are supported.

1. Drag the 'RSS Input' step onto the canvas.
2. Double-click on the step, and configure the following properties:



3. Click on the Fields tab and configure the following properties:
   - Click on the 'Get Fields' button.



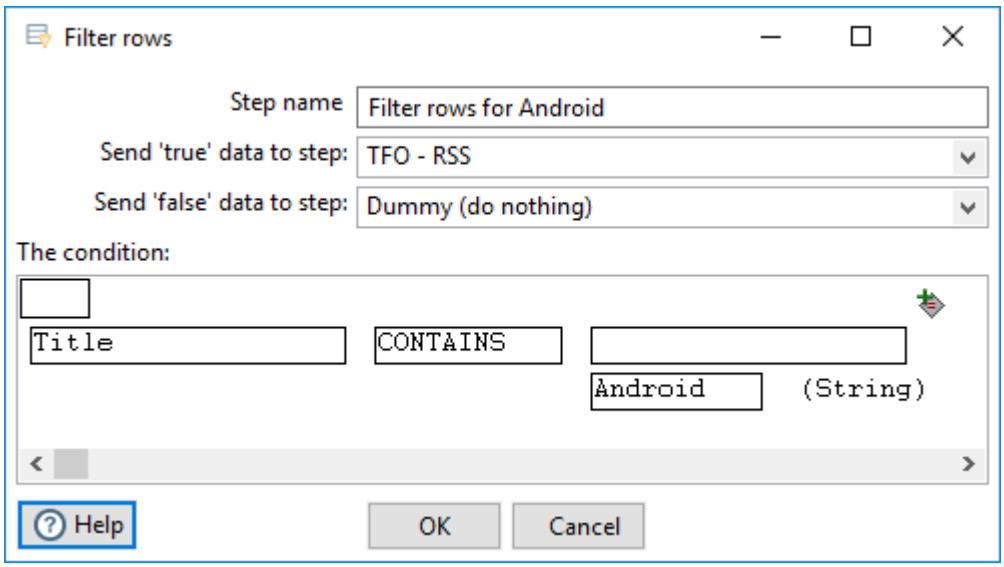
4. Close the step.

Filter

The Filter Rows step allows you to filter rows based on conditions and comparisons. Once this step is connected to a previous step (one or more and receiving input), you can click on the "<field>", "=" and "<value>" areas to construct a condition.

To enter an IN LIST operator, use a string value separated by semicolons. This also works on numeric values like integers. The list of values must be entered with a string type, e.g.: 2;3;7;8

1. Drag the 'Filter rows' step onto the canvas.
2. Double-click on the step, and configure the following properties:



3. Click OK.

- You may need to change the filter value based on Titles published.

Dummy

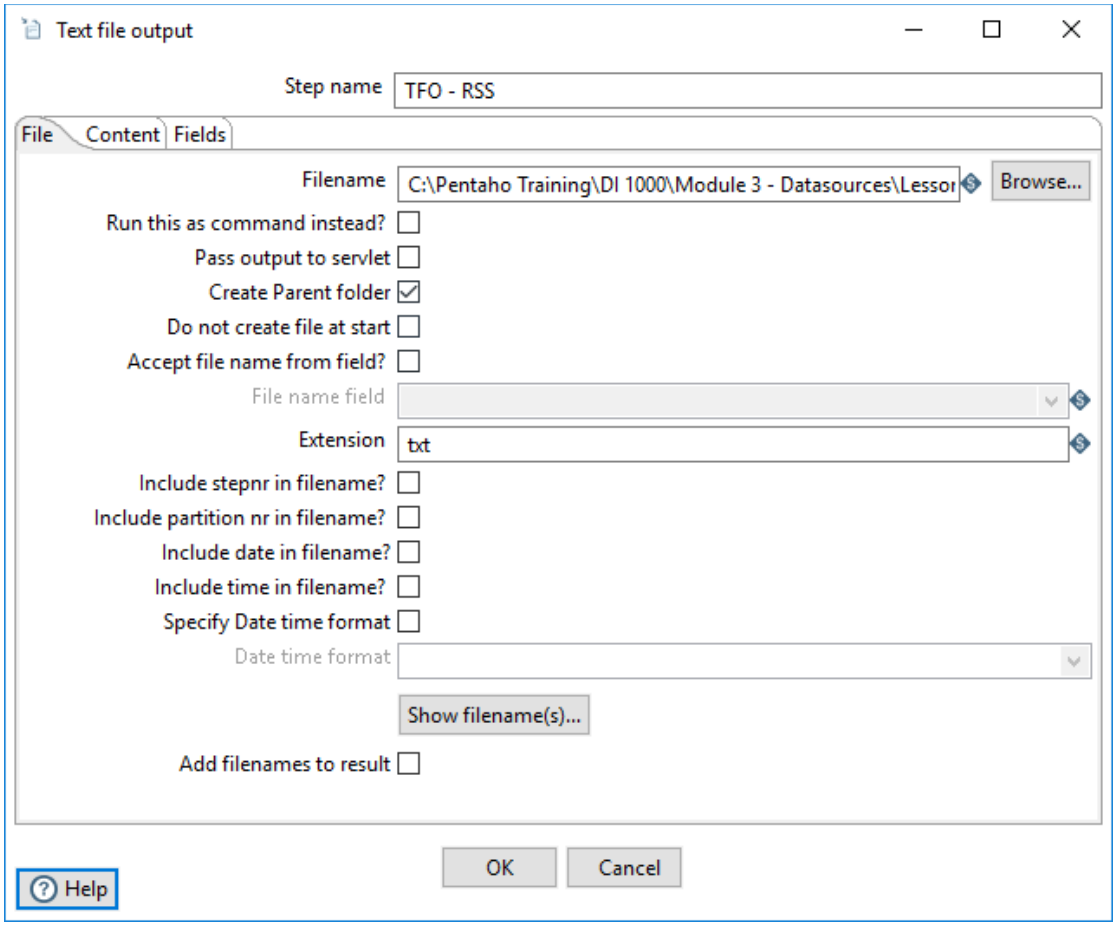
he Dummy step does not do anything. Its primary function is to be a placeholder for testing purposes. For example, to have a transformation, you need at least two steps connected to each other. If you want to test a file input step, you can connect it to a dummy step.

1. Drag the 'Dummy' step onto the canvas.
2. Define the error hop to the step.

Text File Output - RSS

The Text file output step is used to export data to text file format. This is commonly used to generate Comma Separated Values (CSV files) that can be read by spreadsheet applications. It is also possible to generate fixed width files by setting lengths on the fields in the fields tab.

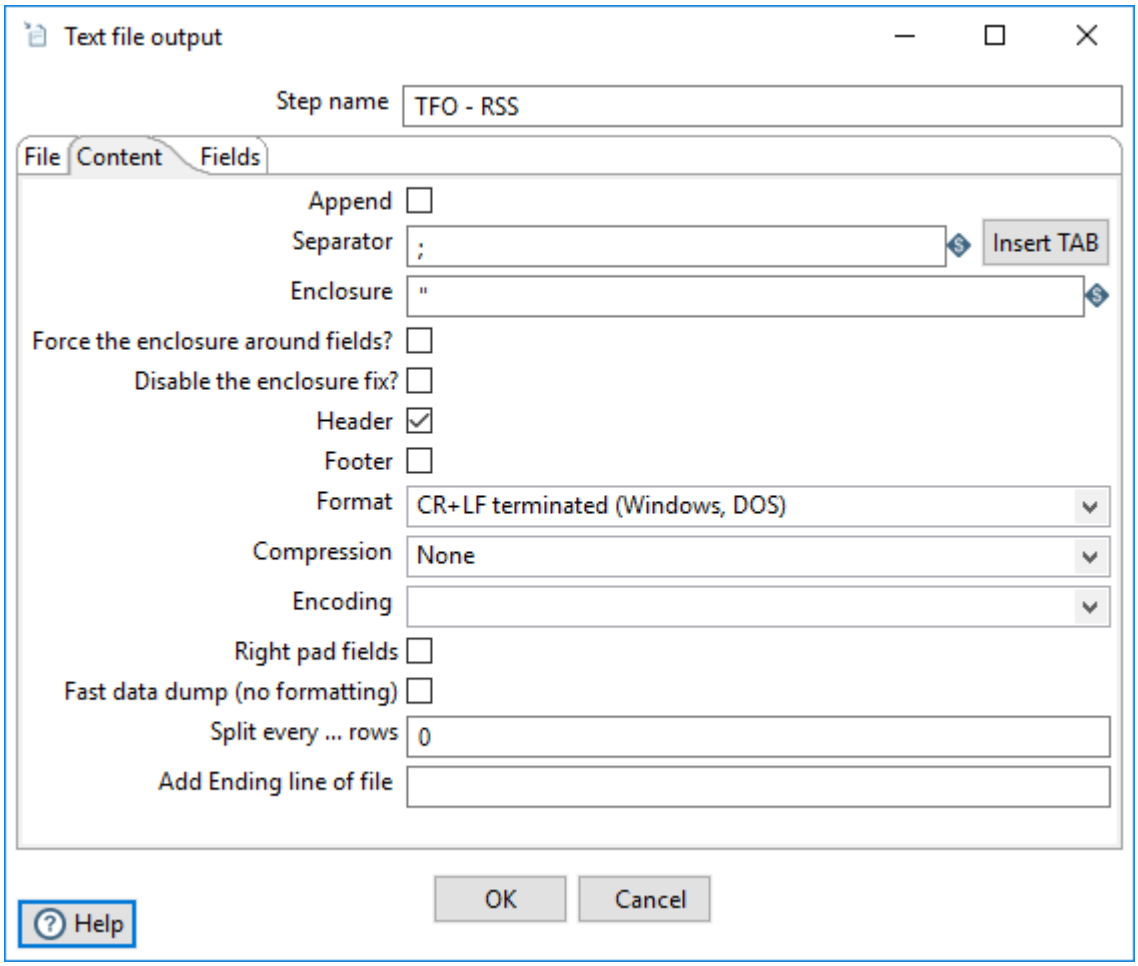
1. Drag the 'Text File Output' step onto the canvas.
2. Double-click on the step, and configure the following properties:



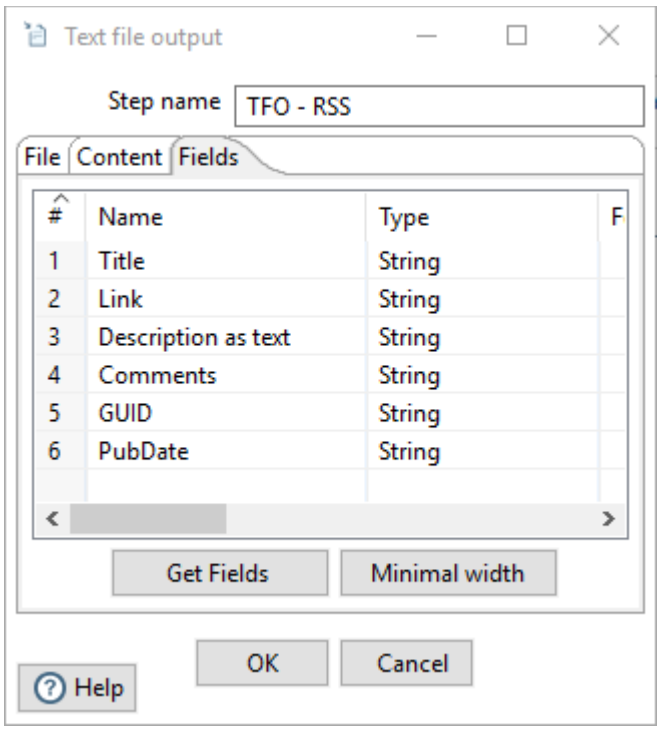
- Point the output file to the relevant training folder:

  File: `C:\Pentaho Training\DI 1000\Module 3 - Datasources\Lesson 1 - Working with Files\Demo 3-1-6 - RSS Feed\rss`

3. Click on the Content tab, and configure as illustrated:

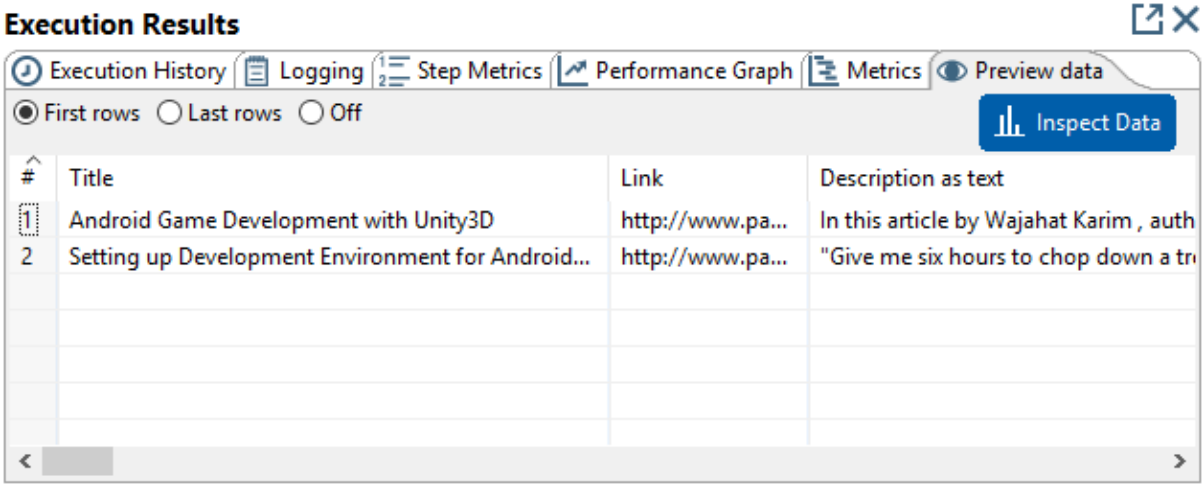


4. Click on the Fields tab, and configure as illustrated:



RUN and Save

1. Click the Run button in the Canvas Toolbar
2. Click on the Text File Output step and Preview data.



Summary

Topics covered in this section:

- Text file as a data source:
    - Text File Input
    - Text File Output
    - Excel Writer
    - Get data from XML
    - JSON
    - RSS Feed

In this lesson, you learned how to get data from files and put data back into files. Specifically, you learned how to:

- Retrieve data from plain files, XML files, JSON and RSS Feeds
- Write data into text files and Excel files
- Get information from the operating system such as command-line arguments and system date

Using internal variables to write location-independent processes

When designing your ETL workflow, using internal variables allow you to set the path to files being read or written too. Once set as variables, you can easily migrate your solution from development to testing to production environments without too much pain..!

PDI has two helpful internal variables, that you can access, by pressing *Ctrl + Space* directly from inside the field.

- Internal.Job.Filename.Directory - This is the directory name where the running job resides
- Internal.Entry.Current Directory - This is the directory name where the running Transformation / File resides (Pentaho v7+ for previous versions .. see below)
- Internal.Transformation.Filename.Directory - This is the directory name where the running transformation resides

The important thing about these two variables is that PDI resolves them dynamically at runtime.

You also configured the following key steps:

Append - This step type allows you to order the rows of two inputs hops.

Block until Step Finish - This step simply waits until all the step copies that are specified in the dialog have finished.

Data Grid - The Data Grid step allows you to enter a static list of rows in a grid.

Excel Writer - The Microsoft Excel Writer step writes incoming rows into an MS Excel file. It supports both the xls and xlsx file formats.

Flattener - The Flattener step allows you flatten data sequentially.

Get Data from XML - This step provides the ability to read data from any type of XML file using XPath specifications.

Get System Info - The Get System Info step retrieves information from the Kettle environment.

JSON Input - The JSON Input step extracts relevant portions out of JSON structures, files or incoming fields, and outputs rows.

RegEx Evaluation - This step type allows you to match the String value of an input field against a text pattern defined by a regular expression. Optionally, you can use the regular expression step to extract substrings from the input text field matching a portion of the text pattern into new output fields. This is known as "capturing".

Replace in String - Replace in string is a simple search and replace. It also supports regular expressions and group references.

Text File Input - The Text File Input step is used to read data from a variety of different text-file types. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

User Defined Java Expression - This step allows you to enter User Defined Java Expressions as a basis for the calculation of new values.

In the next lesson, you will look at working with the Steel Wheels sample database.

Data Sources: Working with Databases

Topics covered in this section:

- Overview of Steel Wheels Database
- Connecting to databases
- Previewing and getting data from a database
- Inserting, updating, and deleting data from a database

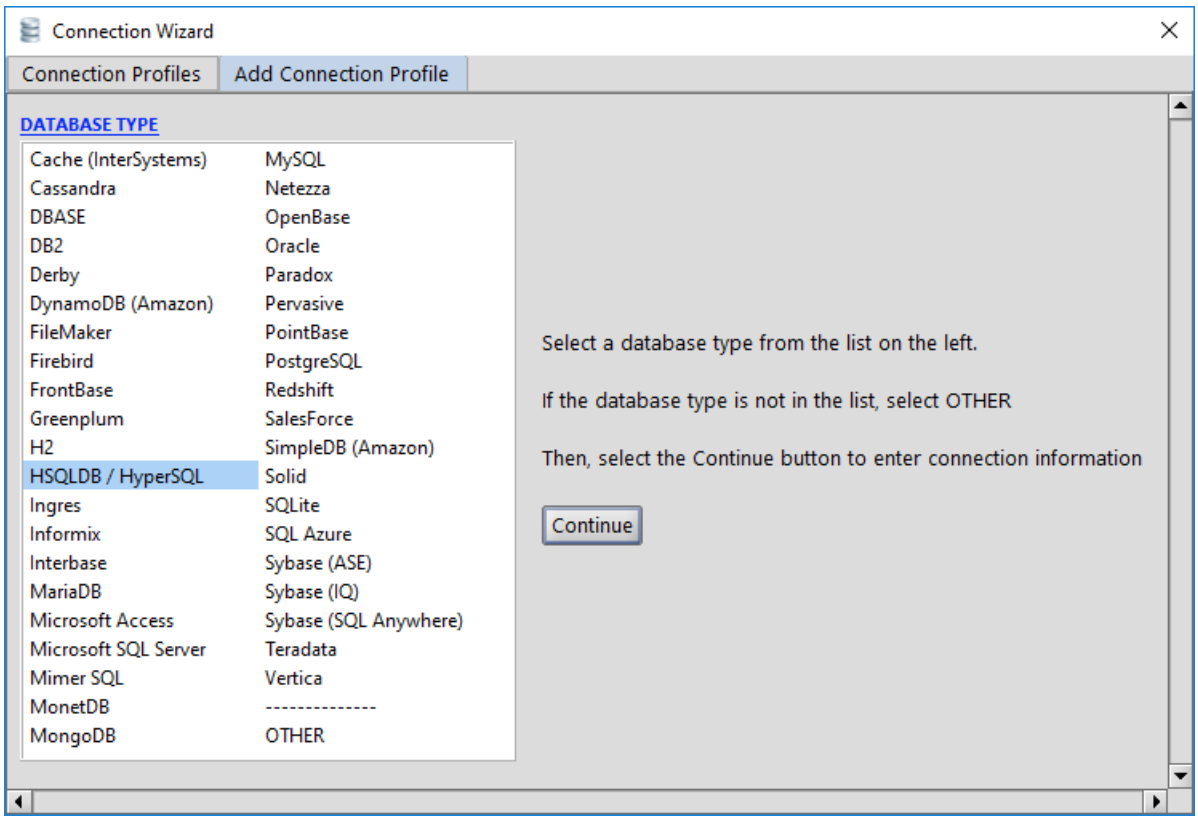
Usually you start working with a database by creating a connection to it. The tool you will be using is RazorSQL v7.1.4; which helps you quickly to create connections to various databases, as the drivers are shipped with the product.

http://downloads.razorsql.com/downloads/7_1_4/razorsql7_1_4_setup_x64.exe

Once you have a database connection you can search for database objects in the Database Navigator, or use the search tools to find specific objects, or compare databases and their contents. You can also edit data and import and export data, and you can create reports about the database and objects in it.

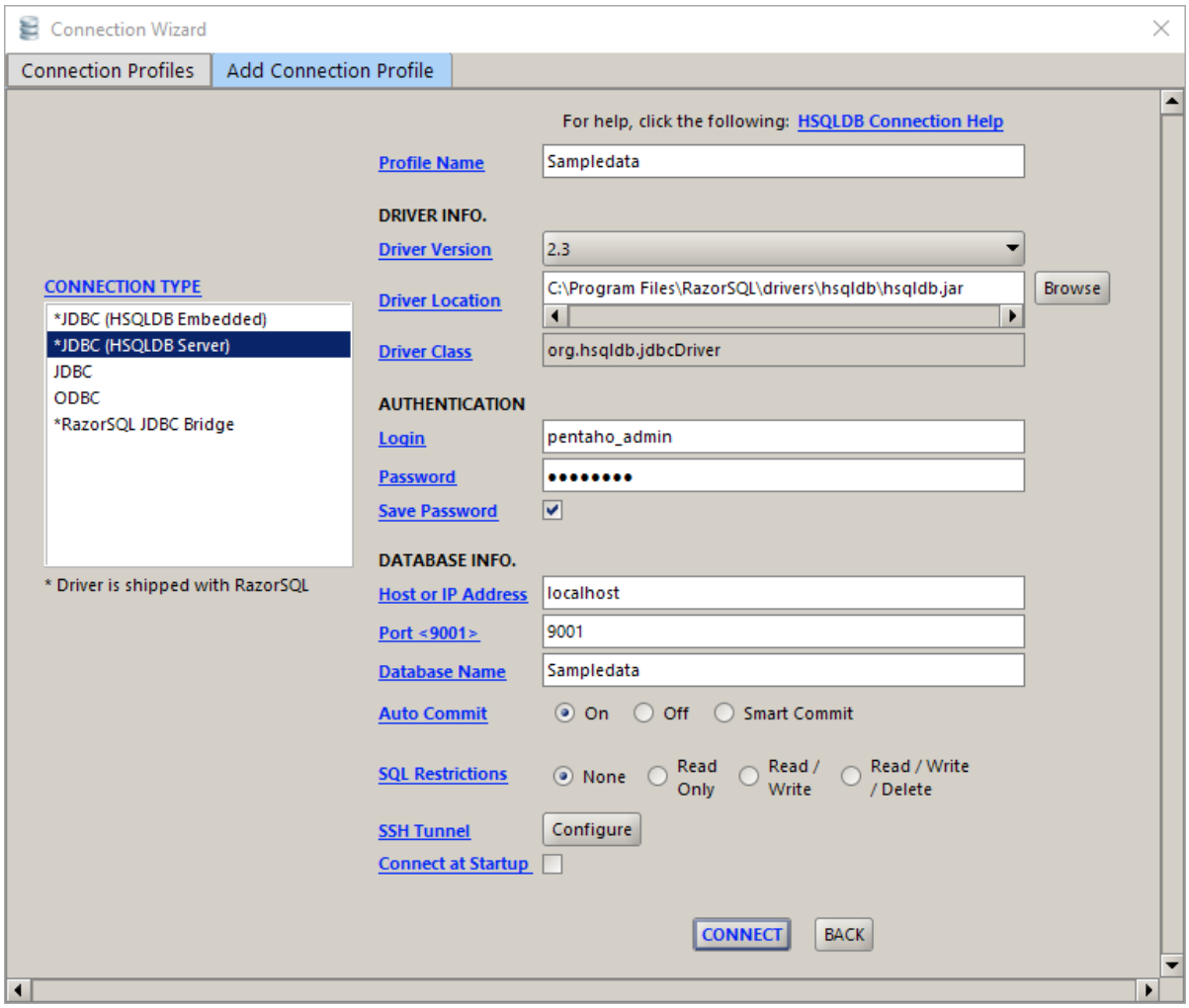
To connect to the SampleData Hypersonic Database

1. Click on: 'Click to connect to a database'.
2. Select HSQLB / HyperSQL, then Continue.



3. Configure the connection as illustrated below:

- Username: pentaho_admin
- Password: password



4. Click CONNECT.

Tip: When you connect, and receive an EOF error message:

- try changing the driver to an earlier version
- change the driver location to:
- C:\Pentaho\design-tools\data-integration\lib\hsqldb-version.jar

Overview of Steel Wheels Database

Before beginning to work on databases, let's briefly introduce the Steel Wheels database along with some database definitions.

The sample Steel Wheels database is a collection of items stored in tables. Typically, all items stored in a table belong to a particular data type. The following table lists some of the tables in the Steel Wheels database:

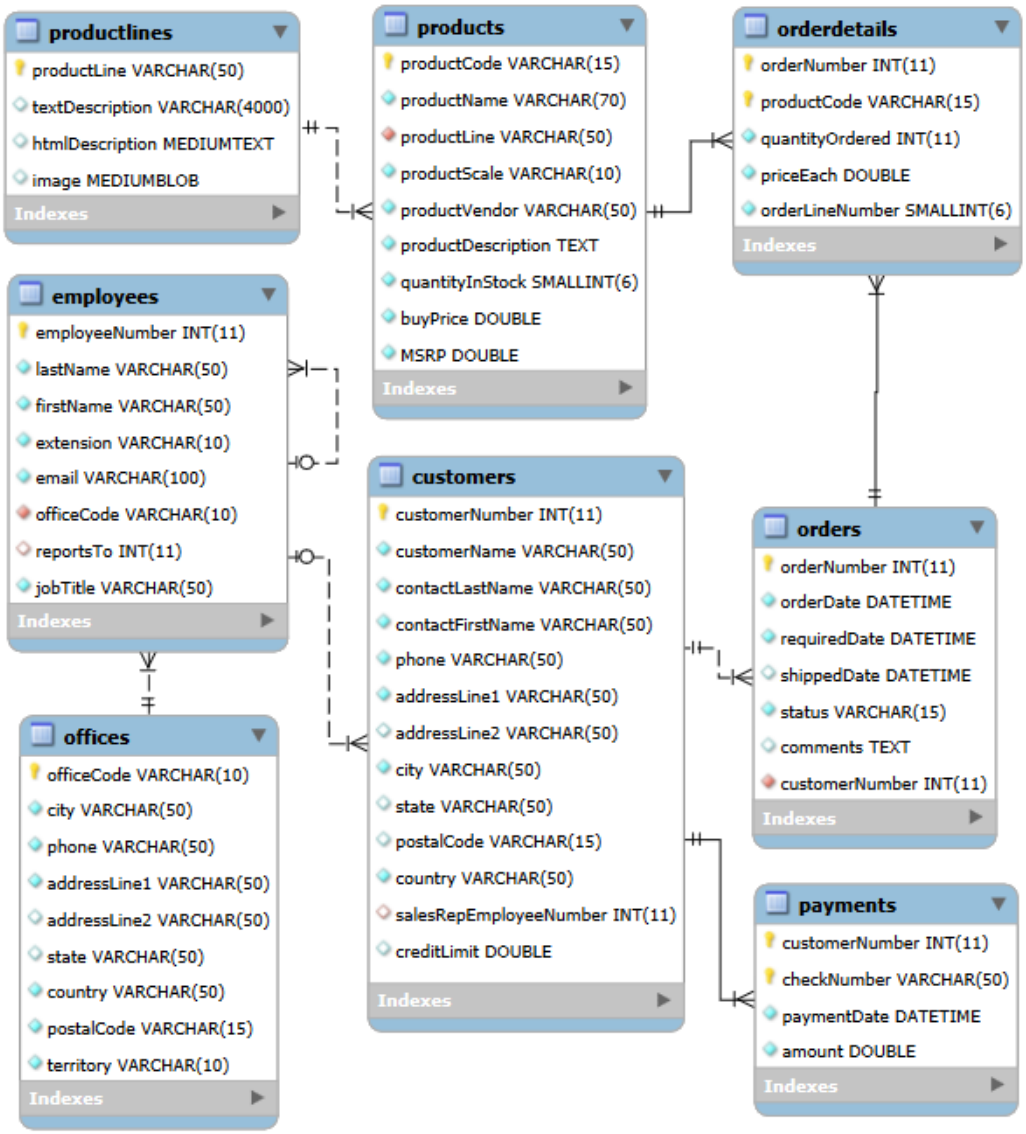


Table	Description
CUSTOMERS	Steel Wheels' customers
EMPLOYEES	All employee information, organization structure such as who reports to whom
PRODUCTS	Products sold by Steel Wheels
PRODUCTLINES	List of product line categories.
OFFICES	Steel Wheels' offices
ORDERS	Information about sales orders
ORDERDETAILS	Sales order line items for each sales order.
PAYMENTS	Payments made by customers based on their accounts.

The items stored in the tables represent an entity or a concept in the real world. As an example, the CUSTOMERS table stores items representing customers. The ORDERS table stores items that represent sales orders in the real world.

In technical terms, a table is uniquely identified by a name such as CUSTOMERS, and contains columns and rows of data.

You can think of a table as a PDI dataset. You have fields (the columns of the table) and rows (the records of the table).

The columns, just like the fields in a PDI dataset, have a metadata describing their name, type, and length. The records hold the data for those columns; each record represents a different instance of the items in the table. As an example, the table CUSTOMERS describe the customers with the columns CUSTOMERNUMBER, CUSTOMERNAME, CONTACTLASTNAME and so forth. Each record of the table CUSTOMERS belongs to a different Steel Wheels' customer.

A table usually has a primary key. A **primary key** or **PK** is a combination of one or more columns that uniquely identify each record of the table. In the sample table, CUSTOMERS, the primary key is made up of a single column—CUSTOMERNUMBER. This means there cannot be two customers with the same customer number.

Tables in a relational database are usually related to one another. For example, the CUSTOMERS and ORDERS tables are related to convey the fact that real-world customers have placed one or more real-world orders. In the database, the ORDERS table has a column named CUSTOMERNUMBER with the number of the customer who placed the order. As said, CUSTOMERNUMBER is the column that uniquely identifies a customer in the CUSTOMERS table. Thus, there is a relationship between both tables. This kind of relationship between columns in two tables is called **foreign key** or **FK**.

Connecting to Databases

Administration tasks include setting up data connections, importing and exporting content, creating clusters, and configuring logging and monitoring operations.

Pentaho Data Integration (PDI) allows you to make connections in each job and transformation through an input step. Although users can create connections themselves, it is best to set up shared connections for your users so that they can simply select the connection they need from a list. We help you download the correct drivers, choose the connection type, and then create the connection.

Database Connections

To connect to databases, install the driver for your database, as well as define the access protocol and settings. You can choose from these access protocols.

- Native (JDBC): This is a commonly used access protocol. Please see details in the Database Access Protocol Decision Table to ensure you make an informed choice.
- JNDI: This also is a commonly used access type. Please see details in the Database Access Protocol Decision Table to ensure you make an informed choice.
- ODBC: We do not support ODBC, and it is our recommendation that you use the JDBC driver instead the ODBC driver. You should only use ODBC when there is the need to connect to unsupported databases by using the generic database driver or other specific reasons. For more information, see Why Should You Avoid ODBC? on the Pentaho public wiki.
- OCI: If you are connecting to an Oracle database, you must first install the appropriate OCI driver and add the OCI connection.

Define JNDI Connections for the DI Server

Pentaho has supplied a way of configuring a JNDI connection for "local" Pentaho Data Integration use so that you do not have an application server continuously running during the development and testing of transformations. To configure, edit the properties file called jdbc.properties located at …\data-integration-server\pentaho-solutions\system\simple-jndi.

> It is important that the information stored in jdbc.properties mirrors the content of your application server data sources.

Introduction	If you intend to work with a database, either reading, writing, looking up data, and so on, the first thing you must do is to create a connection to that database.
Objectives	In this guided demonstration, you will… • Connect to Steel Wheels sample database (Hypersonic)

Define Database Connection (Hypersonic)

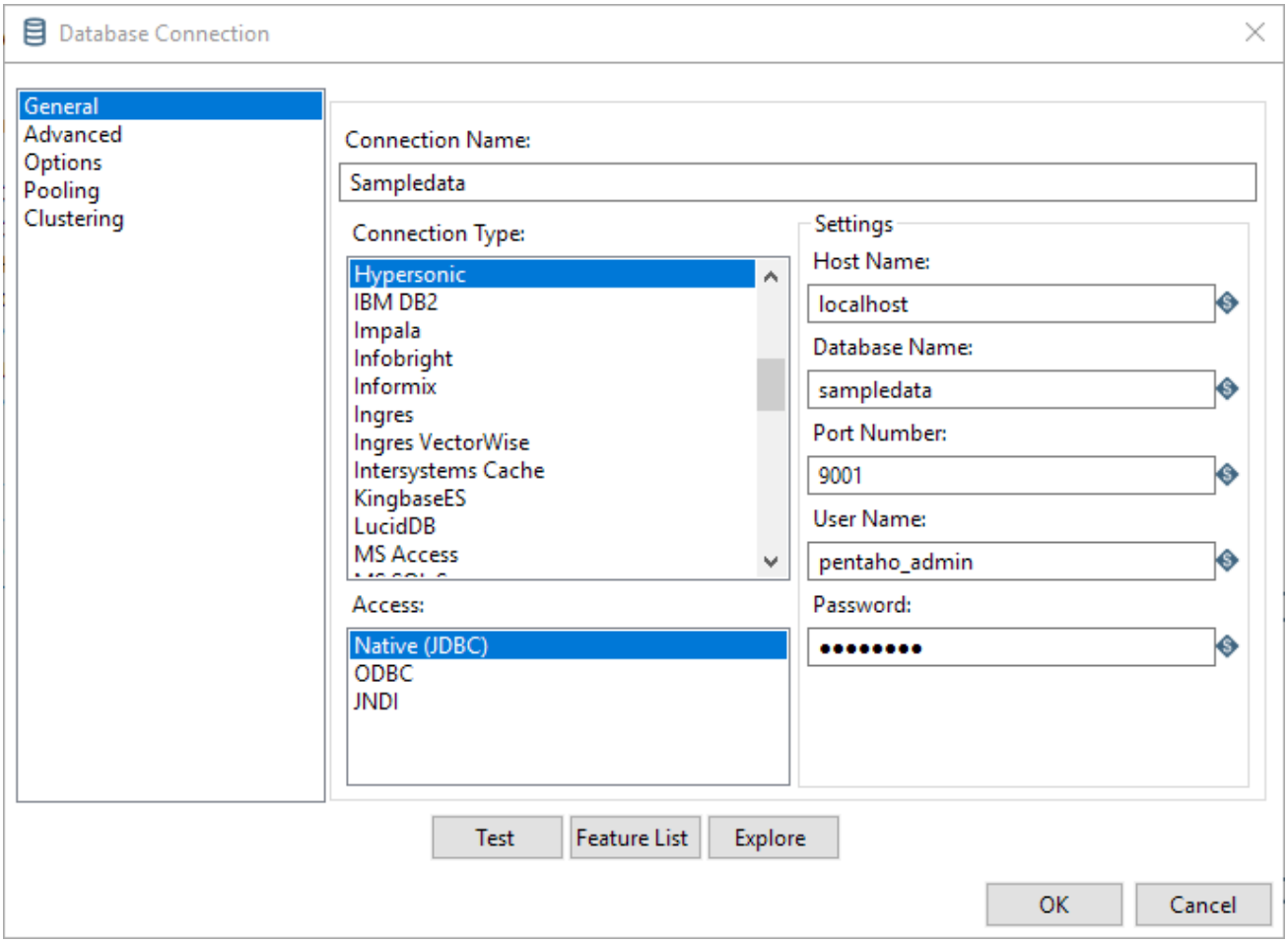
1. In Spoon, click File > New > Transformation.

Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

- By clicking New, then Transformation
- By using the CTRL-N hot key

2. From within Spoon, Select: File > New > Database Connection

   The Database Connection dialog box appears.

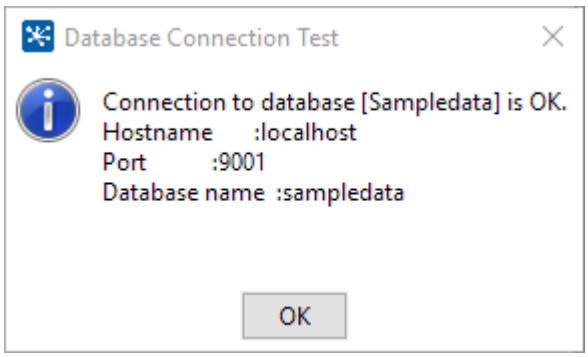


Section Name	What to Do
Connection Name	Type name that uniquely identifies your new connection
Connection Type	Select the type of database to which you are connecting
Access	Select your method of access. Available access types depend on the connecting database type.
Host Name	Type the name of the server that hosts the database to which you are connecting. Alternatively, you can specify the host by IP address.
Database Name	Enter the name of the database to which you are connecting. If you are using a ODBC connection, enter the Data Source Name (DSN) in this field.
Port Number	Enter the TCP/IP port number if it is different from the default.
User name	Optionally, type the user name used to connect to the database.
Password	Optionally, type the password used to connect to the database.

3. Enter the following details:

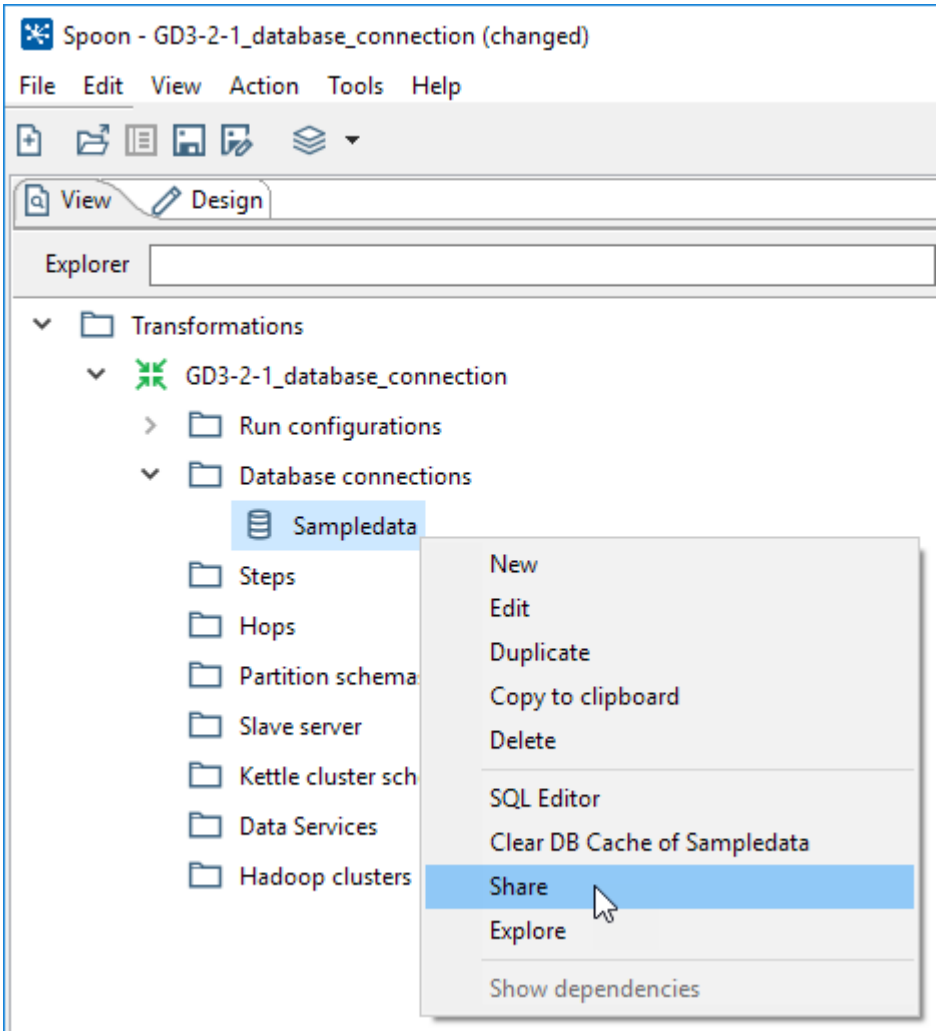
Section Name	Value
Connection Name	Sampledata
Connection Type	Hypersonic
Host Name	Localhost
Database Name	sampledata
User Name	Pentaho_admin
Password	password

4. Click Test. A confirmation message displays if Spoon can establish a connection with the target database.



5. Click OK to save your entries and exit the Database Connection dialog box.

6. From within the View tab, right-click on the connection and select Share from the list that appears. This shares the connection with your users. They will be able to select the shared connection. From within the View tab, click Explore to open the Database Explorer for an existing connection. This shows you the schemas and tables inside the connection.

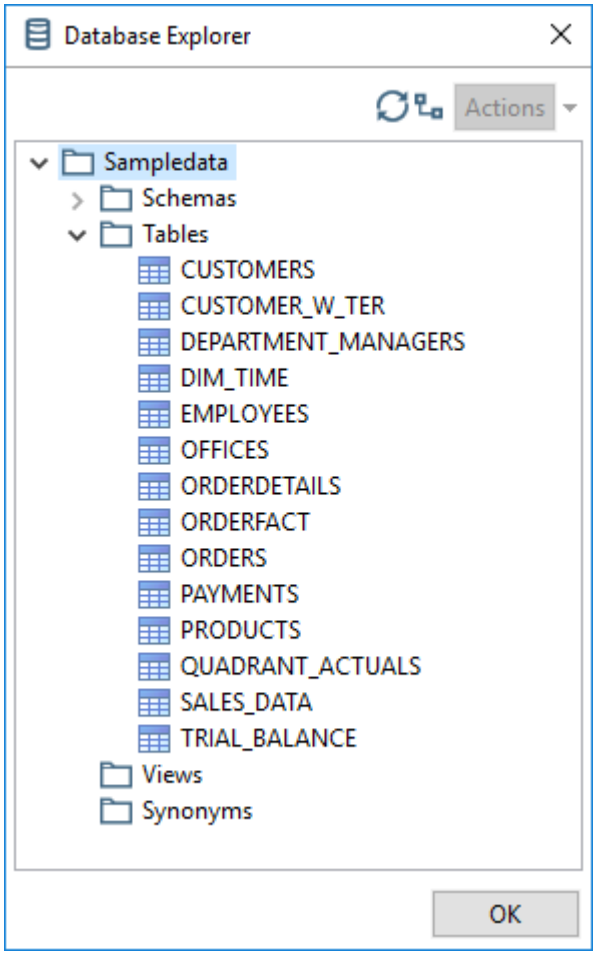


Explore Database

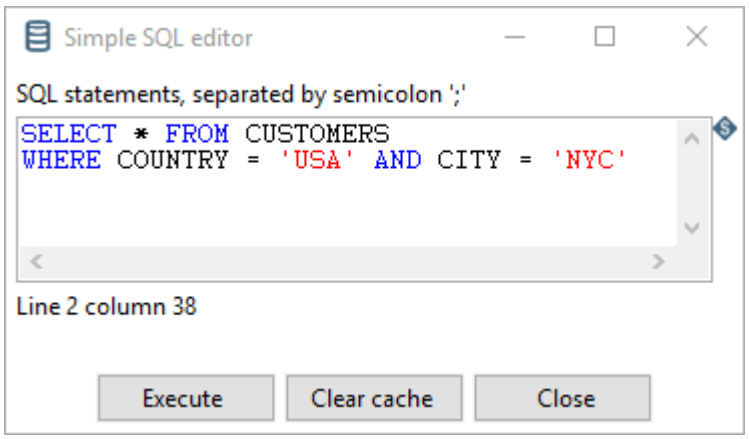
The database explorer allows you to explore any configured database. When you open the database explorer, the first thing you see is a tree with the different objects of the database. As soon as you select a database table, all buttons to the right side become available for you to explore that table. The following are the functions offered by the buttons at the right side of the database explorer:

Option	Action
Preview first 100 rows of …	Return the first 100 rows of the selected table, or all the rows if the table has less than 100. This option shows all columns of the table.
Preview first…rows of …	The same as the previous option, but here you decide the number of rows to show.
Number of rows of …	Tells you the total number of records in the table.
Show layout of …	Shows you the metadata for the columns of the table.
Generate DDL	Shows you the DDL statement that creates the selected table.
Generate DDL for other connection	It lets you select another existent connection. Then it shows you the DDL just like the previous option. The difference is that the DDL is written with the syntax of the database engine of the selected connection.
Open SQL for …	Lets you edit a SELECT statement to query the table. Here you decide which columns and rows to retrieve.
Truncate table	Deletes all rows from the selected table.

1. Click on the View tab, expand Database Connections.
2. Right-click Sampledata and choose Explore from the menu options.
3. In the Database Explorer window, expand Sampledata > Tables



4. Right-click the CUSTOMERS table and choose Preview first 100 from the menu options
5. Examine the customer data.
6. Select 'View SQL'. The simple SQL editor appears. Type this SQL:



7. Click Execute.

Introduction

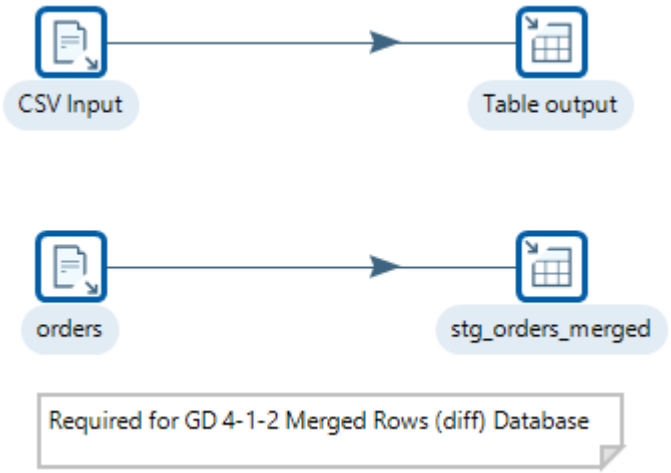
If you work with databases, one of the main objectives will be to extract, load and transform your data. Steel Wheels has several data sources that require loading into a database to discover, cleanse, conform, enrich and validate the data for reports.

Objectives

In this guided demonstration, you will:

- Connect to a database
- Create a database table
- Configure:
  - Table Input

Transformation



CSV File Input

This step provides the ability to read data from a delimited file. The CSV label for this step is a misnomer because you can define whatever separator you want to use, such as pipes, tabs, and semicolons; you are not constrained to using commas. Internal processing allows this step to process data quickly. Options for this step are a subset of the Text File Input step.

This step has fewer overall options than the general Text File Input step, but it has a few advantages over it:

- NIO -- Native system calls for reading the file means faster performance, but it is limited to only local files currently. No VFS support.
- Parallel running -- If you configure this step to run in multiple copies or in clustered mode, and you enable parallel running, each copy will read a separate block of a single file allowing you to distribute the file reading to several threads or even several slave nodes in a clustered transformation.
- Lazy conversion -- If you will be reading many fields from the file and many of those fields will not be manipulate, but merely passed through the transformation to land in some other text file or a database, lazy conversion can prevent Kettle from performing unnecessary work on those fields such as converting them into objects such as strings, dates, or numbers.

1. Drag the CSV file input step onto the canvas.
2. Open the CSV file input properties dialog box.
   Ensure the following details are configured, as outlined below:

Stepname	CSV File Input
Filename	[path to demo folder]\sales_data.csv
Delimiter	Semi-colon;
Lazy conversion	Unchecked
Header row	Checked

3. Click on the Get Fields button.
4. Click OK.

Table Output

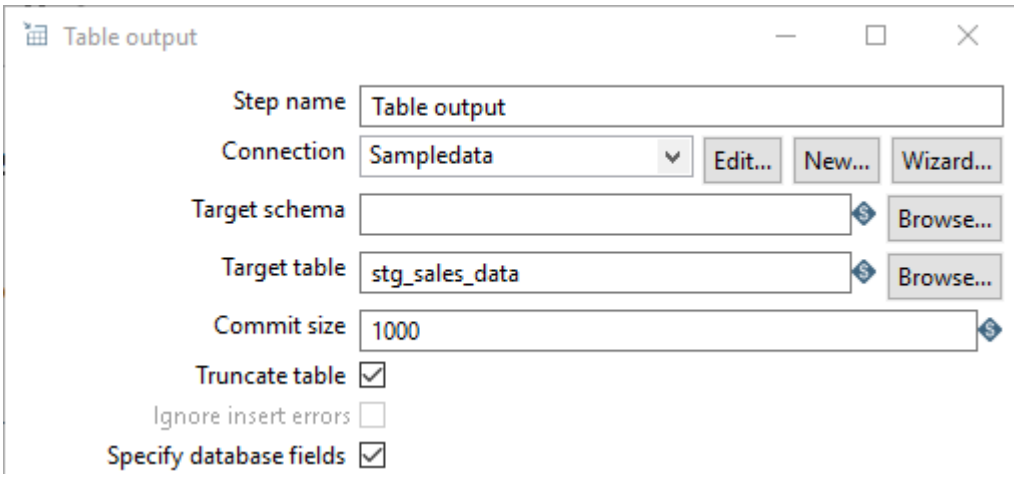
The Table Output step allows you to load data into a database table. Table Output is equivalent to the DML operator INSERT. This step provides configuration options for target table and a lot of housekeeping and/or performance-related options such as Commit Size and Use batch update for inserts.

If you have a database table that has identity columns and you are inserting a record, as part of the insert, the JDBC driver will typically return the auto-generated key it used when performing the insert.

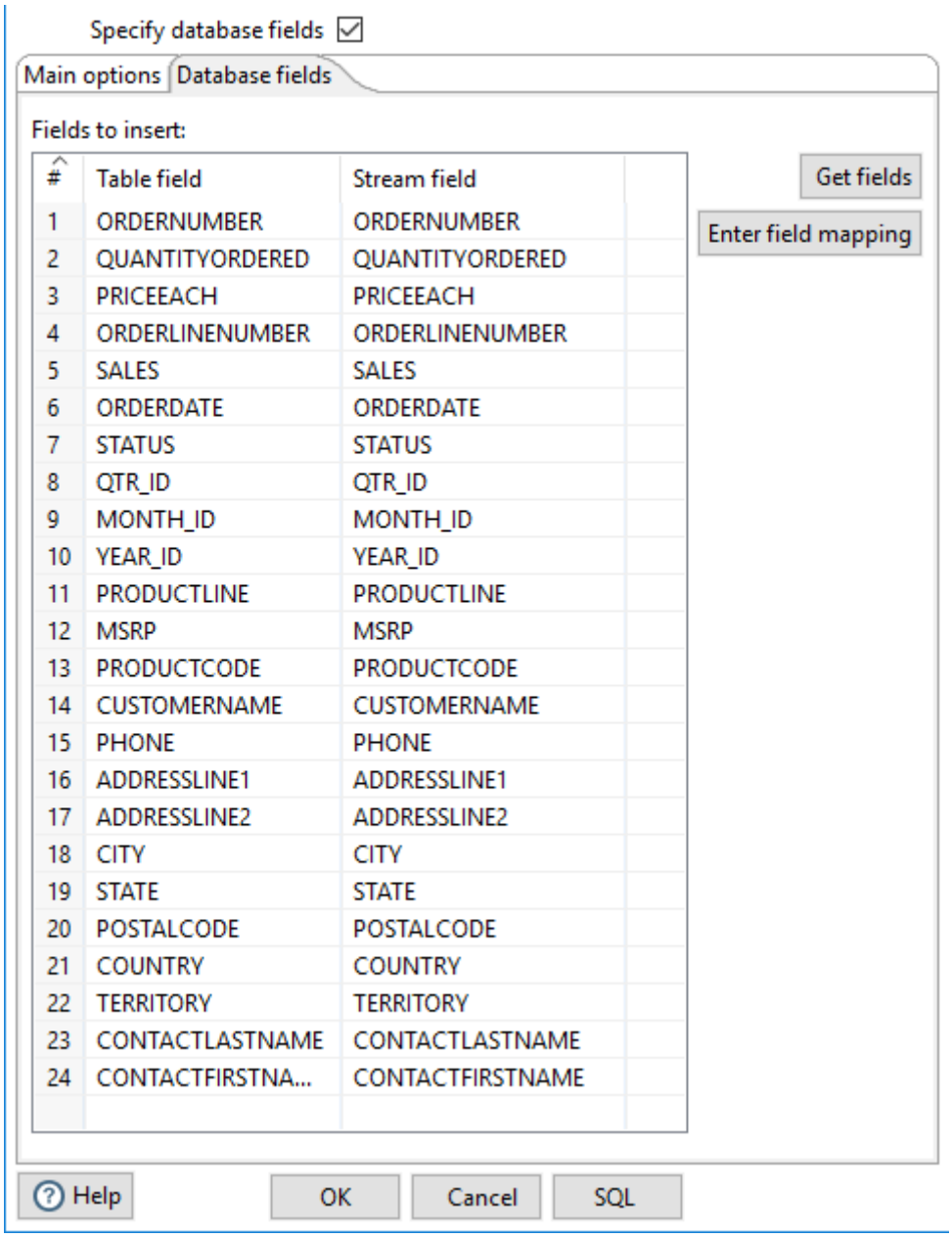
Note: This is not supported on all database types.

Performance notes: There are often performance settings specific to a database type that can be set within the database connection JDBC properties. Please see your database manual for specific JDBC performance settings.

1. Drag the Table Output step onto the canvas.
2. Open the Table Output properties dialog box.
   Ensure the following details are configured, as outlined below:

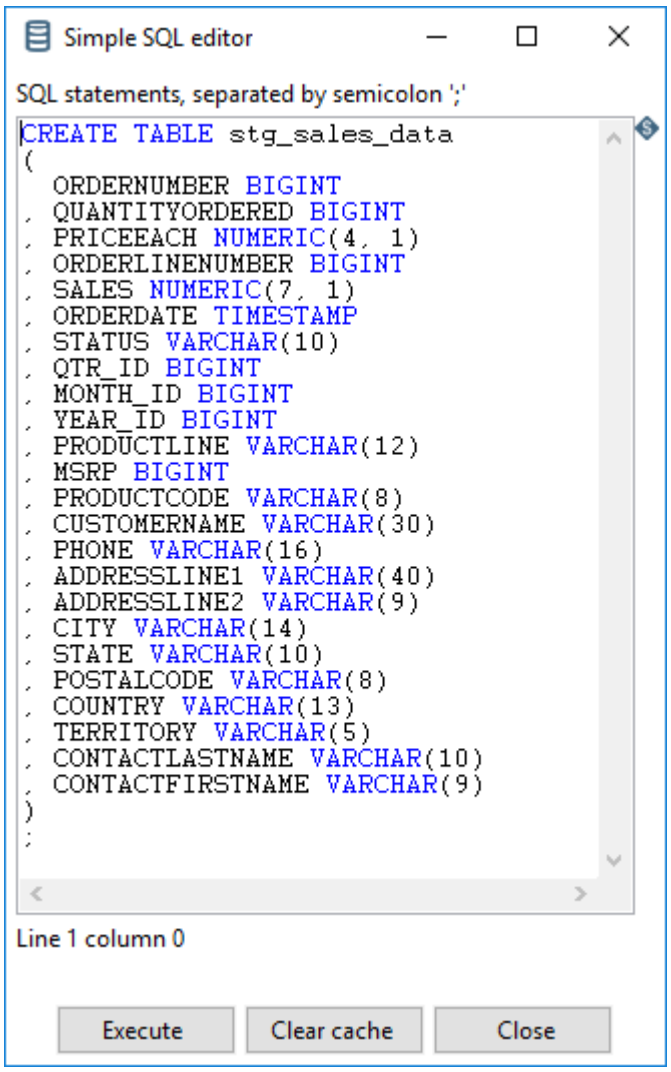


3. Click on the Database fields.
4. Click on the 'Get Fields' button.



5. Click on the SQL button

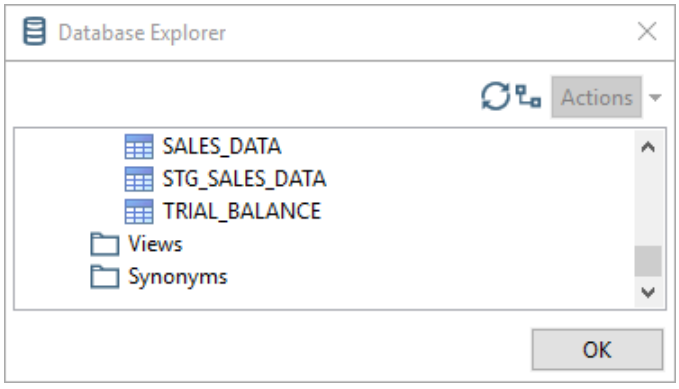
- This will create the required SQL script for the table



6. Click Execute.
7. Click OK to Close all windows.

RUN the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Check that the table has been created in the SampleData database.



Guided Demo 3-2-3: Reading from a Database Table

Introduction

The warehouse manager at Steel Wheels requires a report highlighting the status of the ORDERS shipped.

Objectives

In this guided demonstration, you will:

- Connect to a database
- Modify SQL statement
- Configure the following steps:
  - Table Input
  - Calculator
  - Number Ranges
  - Sort Rows

Transformation

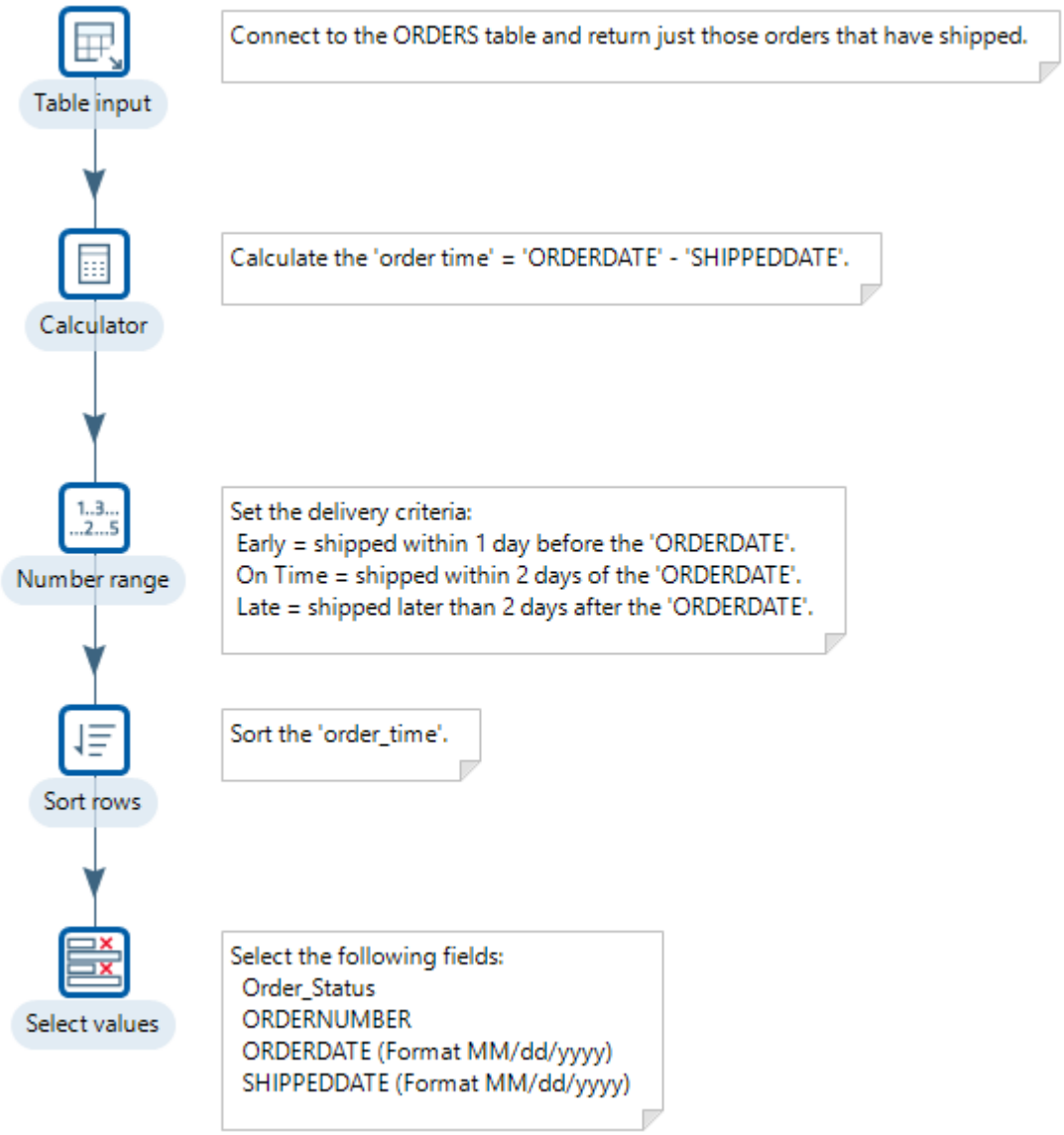
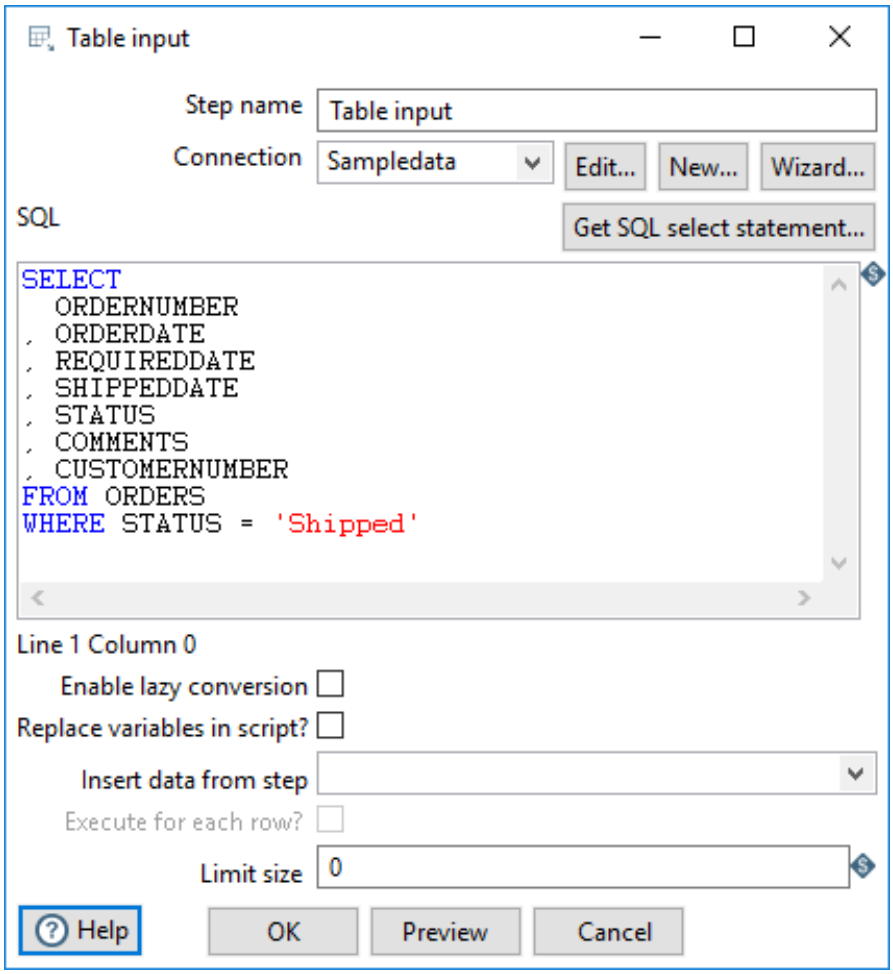


Table Input

This step is used to read information from a database, using a connection and SQL. Basic SQL statements can be generated automatically by clicking ***Get SQL select statement***.

1. Drag the Table Input step onto the canvas.
2. Open the Table Input properties dialog box.
   Ensure the following details are configured, as outlined below:



3. Preview and Click OK

- Connects to the ORDERS data table and extracts the required dataset where the status of the order is 'Shipped'.

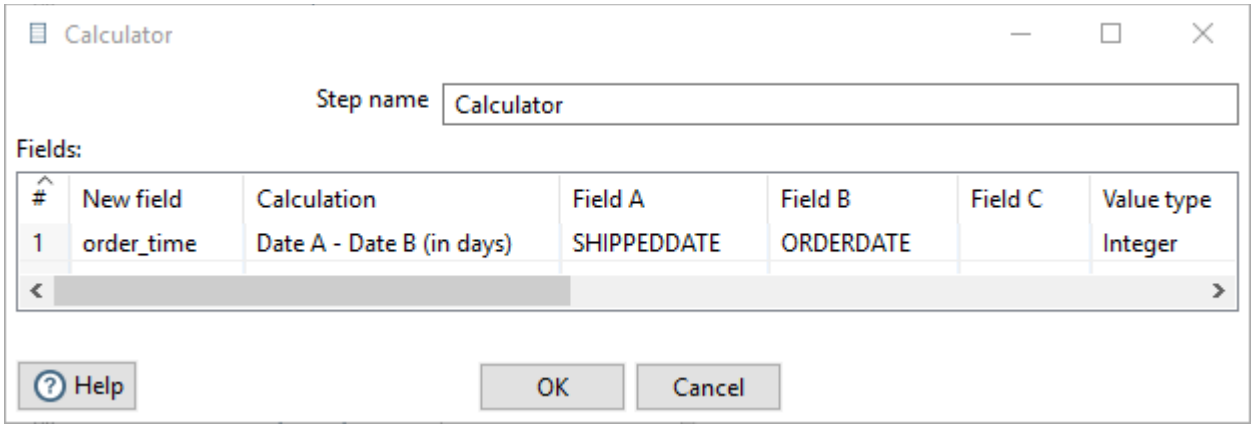
Calculator

This calculator step provides you with predefined functions that can be executed on input field values. If need other generic, often used functions, visit the Pentaho community page and let Pentaho know about your enhancement request.

Note: The execution speed of the Calculator is far better than the speed provided by custom scripts (JavaScript).

Besides the arguments (Field A, Field B and Field C) you must also specify the return type of the function. You can also choose to remove the field from the result (output) after all values are calculated; this is useful for removing temporary values.

1. Drag the Calculator step onto the canvas.
2. Open the Calculator properties dialog box.
3. Ensure the following details are configured, as outlined below:



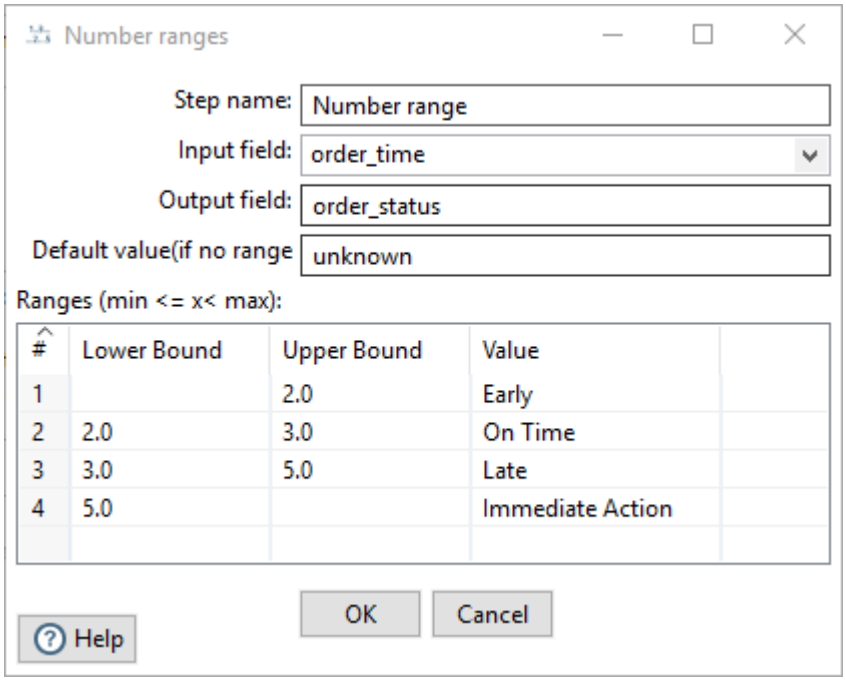
4. Click OK.

- Calculates `order_time` between when the order is required from when it was shipped.

Number ranges

Create ranges based on numeric fields.

1. Drag the Number ranges step onto the canvas.
2. Open the Number ranges properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click OK.

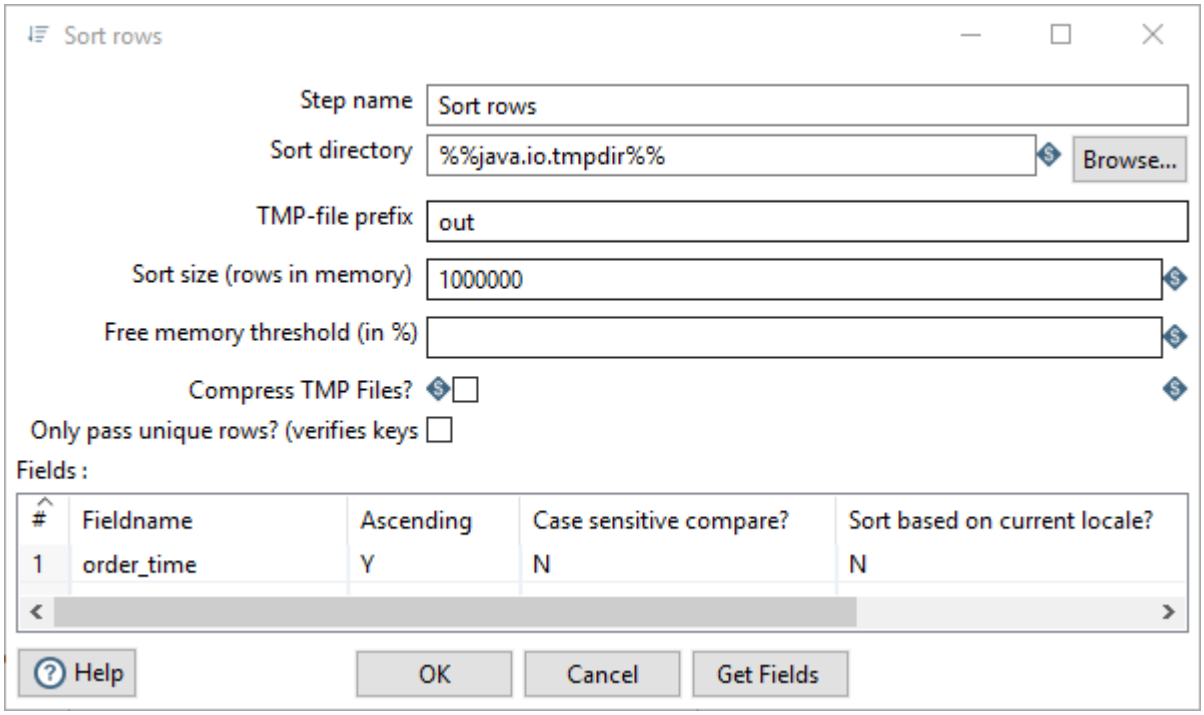
- Sets the actions for '`order_time`' in the output field '`order_status`':
    - Early - If the order has been Shipped more than 2 days or more before the ORDERDATE.
    - On Time - If the order has been Shipped within 2 days or more before the ORDERDATE.
    - Late - If the order has been Shipped more than 2 days and before 5 days of the ORDERDATE.
    - Immediate Action – if the order has been shipped later than 5 days after the ORDERDATE.

Sort Rows

The Sort rows step sorts rows based on the fields you specify and on whether they should be sorted in ascending or descending order.

- Kettle must sort rows using **temporary files** when the number of rows exceeds the specified **sort size** (default 1 million rows). When you get an out of memory exception (OOME), you need to lower this size of change your available memory.
- When you use **multiple copies of the step** in parallel (on the local JVM with "Change number of copies to start" or in a clustered environment using Carte) each of the sorted blocks need to be merged together to ensure the proper sort sequence. This can be done, be adding the Sorted Merge step afterwards (on the local JVM without multiple copies to start or in the cluster on the master).

1. Drag the Sort Rows step onto the canvas.
2. Open the Sort Rows properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click OK.

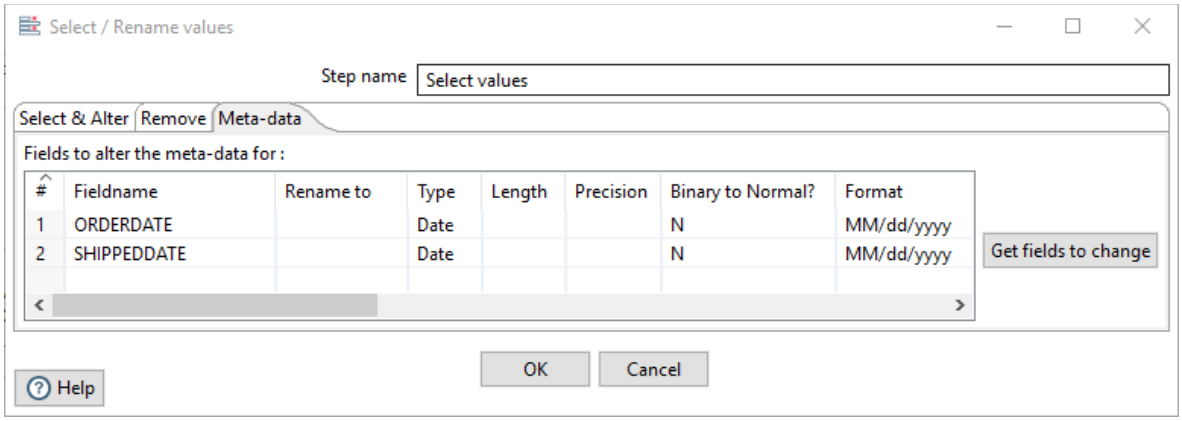
- Before you can do any stream operations, its best practice to Sort the rows.

Select Values

The Select Values step is useful for selecting, removing, renaming, changing data types and configuring the length and precision of the fields on the stream. These operations are organized into different categories:

- **Select and Alter** — Specify the exact order and name in which the fields should be placed in the output rows
- **Remove** — Specify the fields that should be removed from the output rows
- **Meta-data** - Change the name, type, length and precision (the metadata) of one or more fields

1. Drag the Select values step onto the canvas.
2. Open the Select values properties dialog box.
   Ensure the following details are configured, as outlined below:

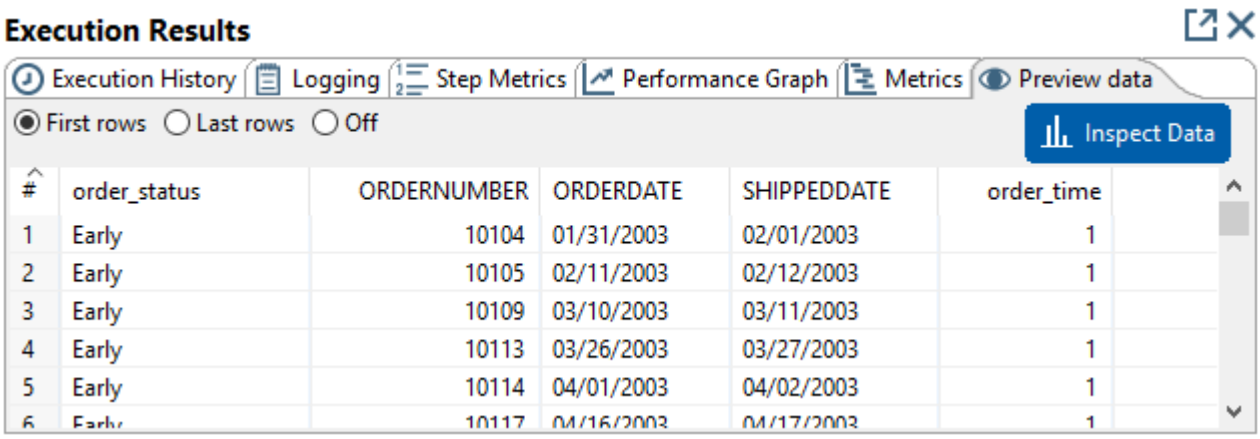


3. Click OK.

- Formats the REQUIREDDATE and SHIPPEDDATE

RUN the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Click the Preview Data tab for Select values.



#	order_status	ORDERNUMBER	ORDERDATE	SHIPPEDDATE	order_time
1	Early	10104	01/31/2003	02/01/2003	1
2	Early	10105	02/11/2003	02/12/2003	1
3	Early	10109	03/10/2003	03/11/2003	1
4	Early	10113	03/26/2003	03/27/2003	1
5	Early	10114	04/01/2003	04/02/2003	1
6	Early	10117	04/16/2003	04/17/2003	1

Guided Demo 3-2-3: Reading from a Database Table (optional)

Introduction

If you work with databases, one of the main objectives will be to extract, load and transform your data.

Steel Wheels wish to launch a marketing campaign, aimed at those customers who haven't placed an order within 30 days.

Objectives

In this guided demonstration, you will:

- Connect to a database
- Modify SQL statement
- Configure the following steps:
  - Table Input
  - Sort Rows
  - Group By
  - Analytic Query
  - Number Ranges
  - Switch Case

Transformation

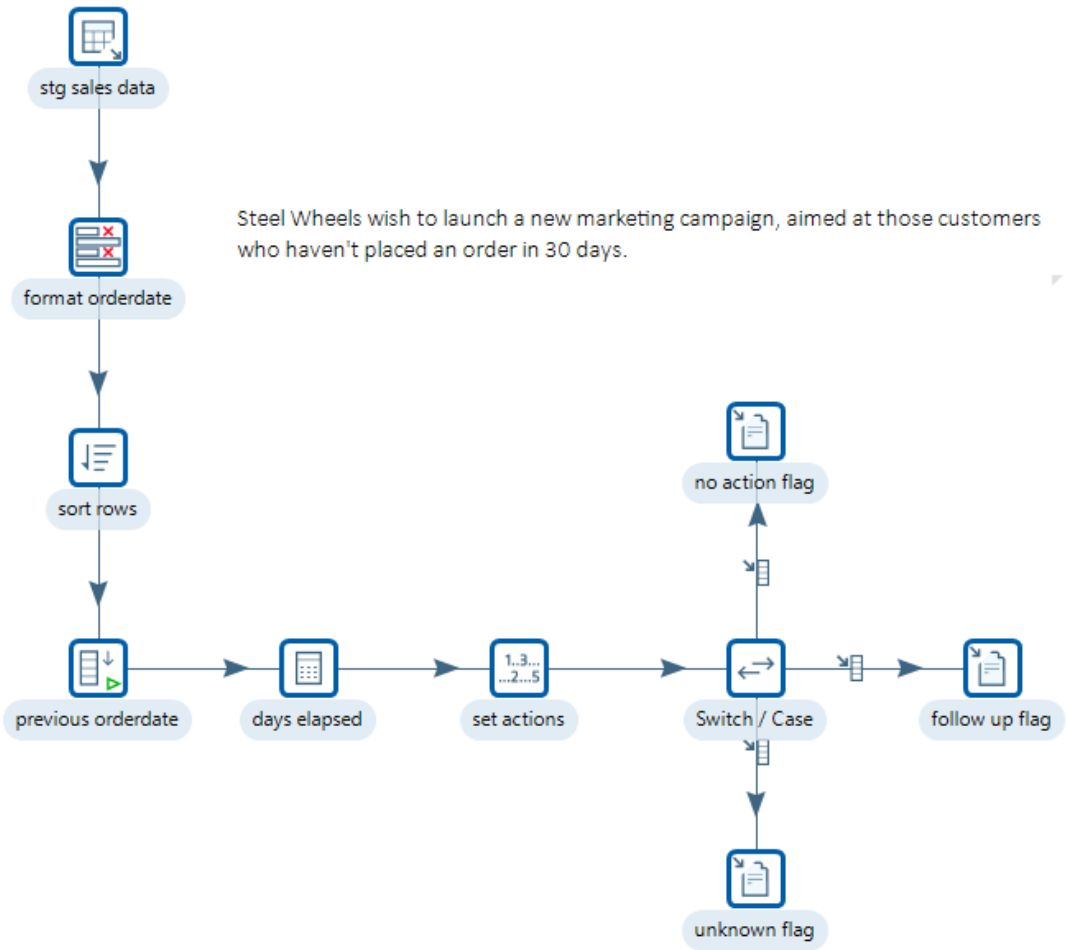
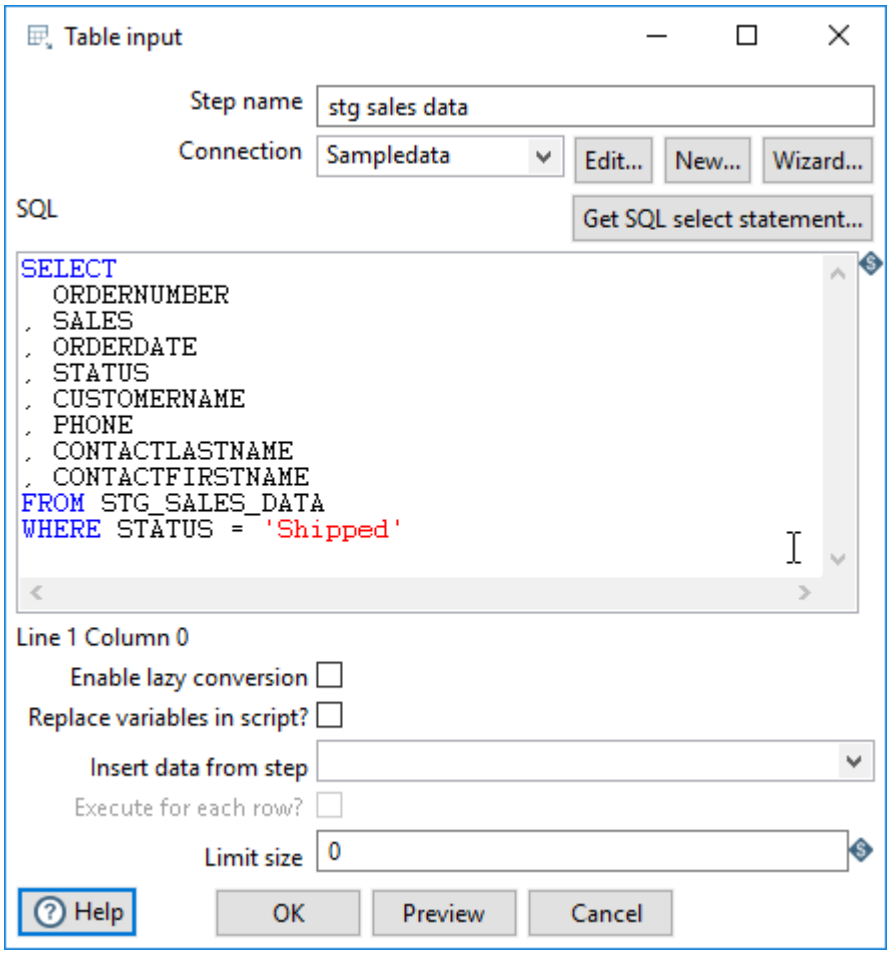


Table Input - stg sales data

This step is used to read information from a database, using a connection and SQL. Basic SQL statements can be generated automatically by clicking ***Get SQL select statement***.

1. Drag the Table Input step onto the canvas.
2. Open the Table Input properties dialog box.
   Ensure the following details are configured, as outlined below:



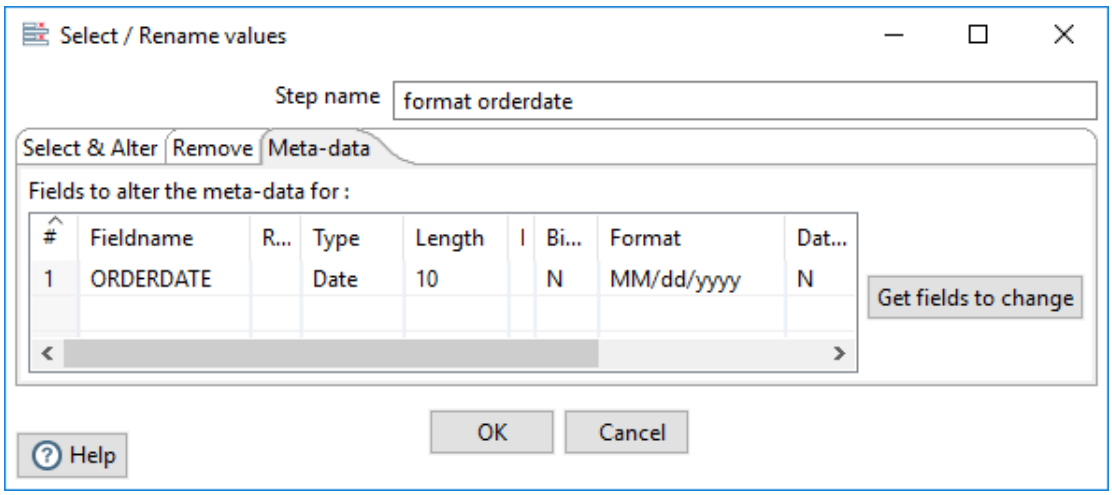
3. Preview and Click OK

- Connects to the stg sales data table and extracts the required dataset where the status of the order is 'Shipped'.

Select Values - Format Orderdate

The Select Values step is useful for selecting, removing, renaming, changing data types and configuring the length and precision of the fields on the stream. These operations are organized into different categories:

- **Select and Alter** — Specify the exact order and name in which the fields should be placed in the output rows
- **Remove** — Specify the fields that should be removed from the output rows
- **Meta-data** - Change the name, type, length and precision (the metadata) of one or more fields
- Drag the Select values step onto the canvas.
- Open the Select values properties dialog box.
  Ensure the following details are configured, as outlined below:



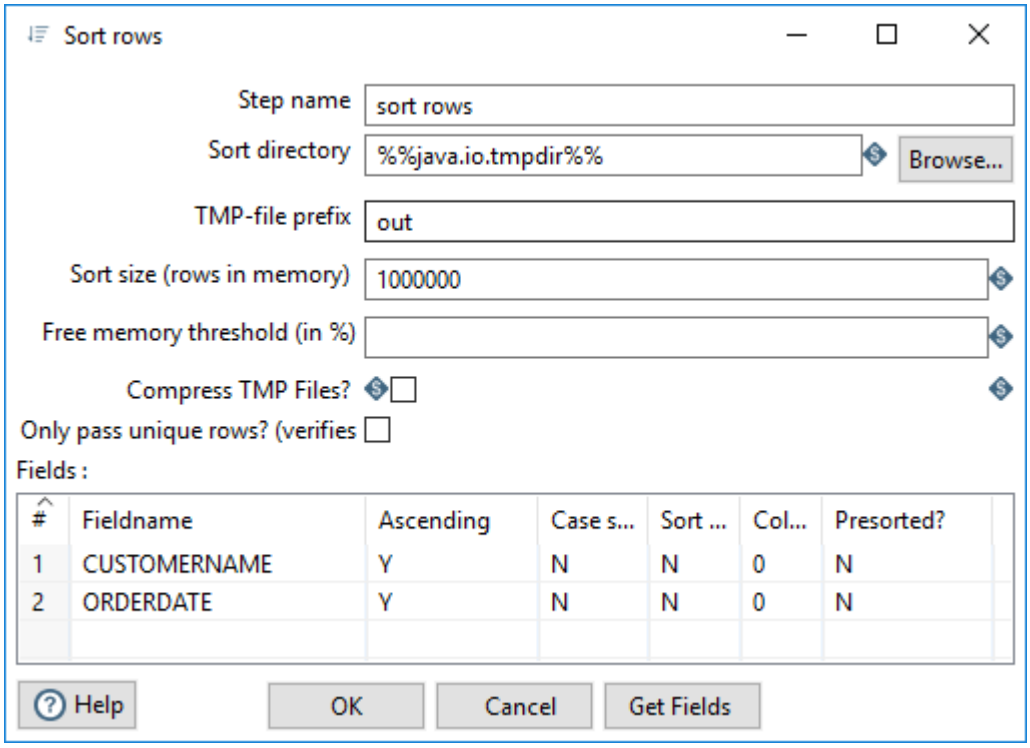
- Click OK.
- Formats the ORDERDATE

Sort Rows

The Sort rows step sorts rows based on the fields you specify and on whether they should be sorted in ascending or descending order.

- Kettle must sort rows using **temporary files** when the number of rows exceeds the specified **sort size** (default 1 million rows). When you get an out of memory exception (OOME), you need to lower this size of change your available memory.
- When you use **multiple copies of the step** in parallel (on the local JVM with "Change number of copies to start" or in a clustered environment using Carte) each of the sorted blocks need to be merged together to ensure the proper sort sequence. This can be done, be adding the Sorted Merge step afterwards (on the local JVM without multiple copies to start or in the cluster on the master).

1. Drag the Sort Rows step onto the canvas.
2. Open the Sort Rows properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click OK.

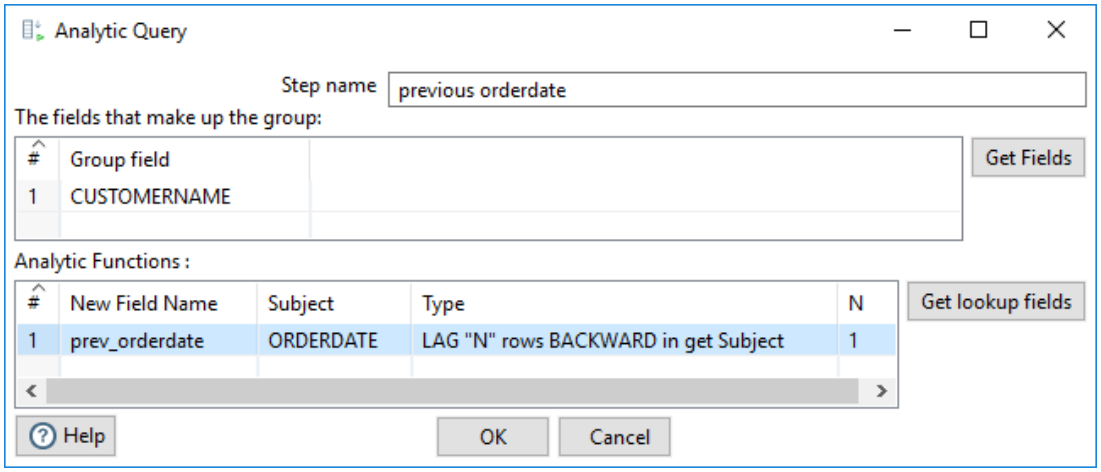
- Before you can do any stream operations, its best practice to Sort the rows.
- Kettle sorts rows using **temporary files** when the number of rows exceeds the specified **sort size** (default 1 million rows). When you get an out of memory exception (OOME), you need to lower this size of change your available memory.

Analytic Query

This step allows you to 'scroll' forward and backwards across rows. Examples of common use cases are:

- Calculates the "time between orders" by ordering rows by Customername, and LAGing 1 row back to get previous order time.
- Calculates the "duration" of a web page view by LEADing 1 row ahead and determining how many seconds the user was on this page.

1. Drag the Group by step onto the canvas.
2. Open the Group by properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click OK.

- The step determines the previous Orderdate for each sorted Customername.

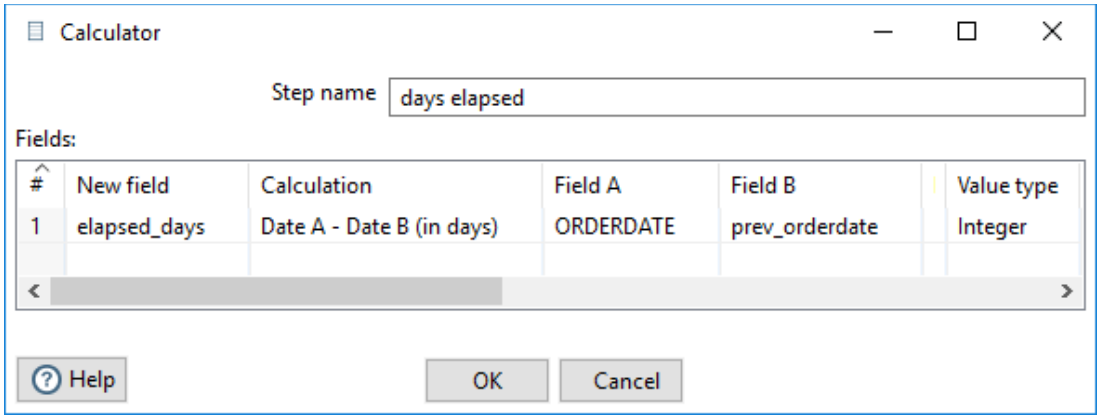
Calculator - Days elapsed

This calculator step provides you with predefined functions that can be executed on input field values. If need other generic, often used functions, visit the Pentaho community page and let Pentaho know about your enhancement request.

Note: The execution speed of the Calculator is far better than the speed provided by custom scripts (JavaScript).

Besides the arguments (Field A, Field B and Field C) you must also specify the return type of the function. You can also choose to remove the field from the result (output) after all values are calculated; this is useful for removing temporary values.

1. Drag the Calculator step onto the canvas.
2. Open the Calculator properties dialog box.
3. Ensure the following details are configured, as outlined below:



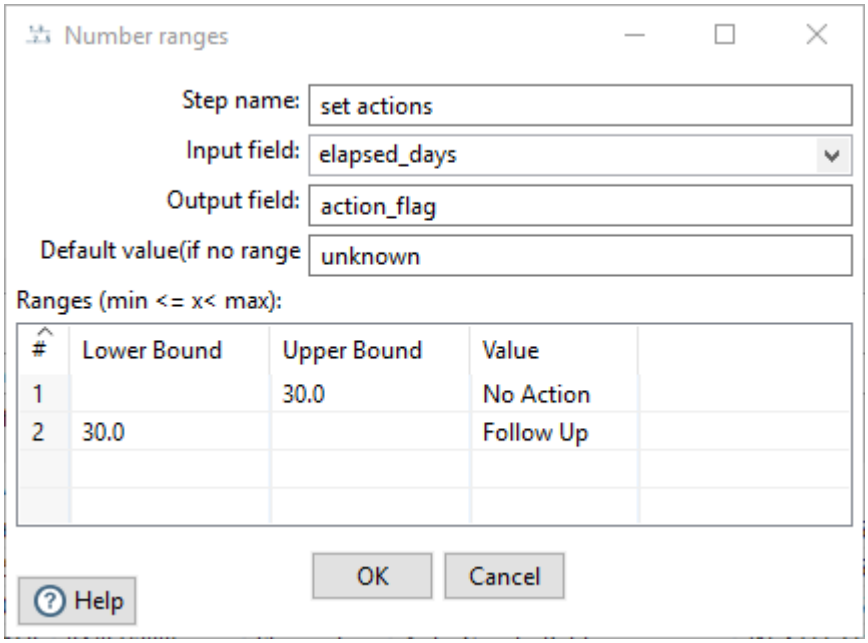
4. Click OK.

- Calculates elapsed_days between orders for each customer.

Number ranges - Set Actions

Create ranges based on numeric fields.

1. Drag the Number ranges step onto the canvas.
2. Open the Number ranges properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click OK.

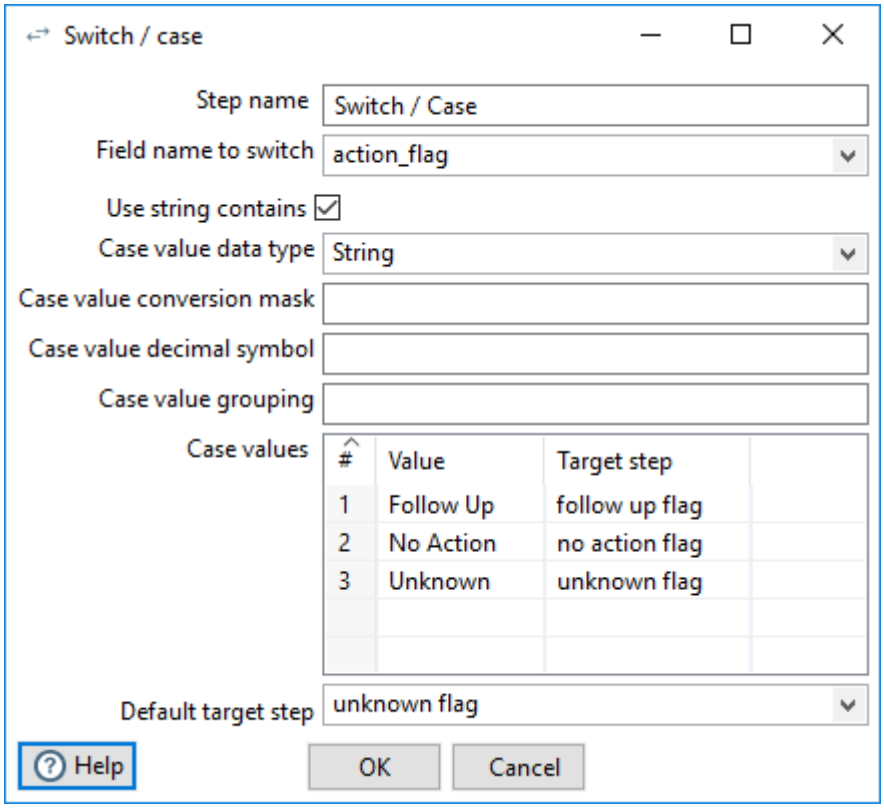
- Sets the actions for '100 elapsed_days':
    - No Action - less than 100 elapsed_days between orders.
    - Follow Up – more than 100 elapsed_days between orders.

Switch Case

What this step does is implement the Switch/Case statement found in popular programming languages like Java.

In our case, we route rows of data to one or more target steps based on the value of the action_flag stream field.

1. Drag the Switch Case step onto the canvas.
2. Open the Switch Case properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click OK.

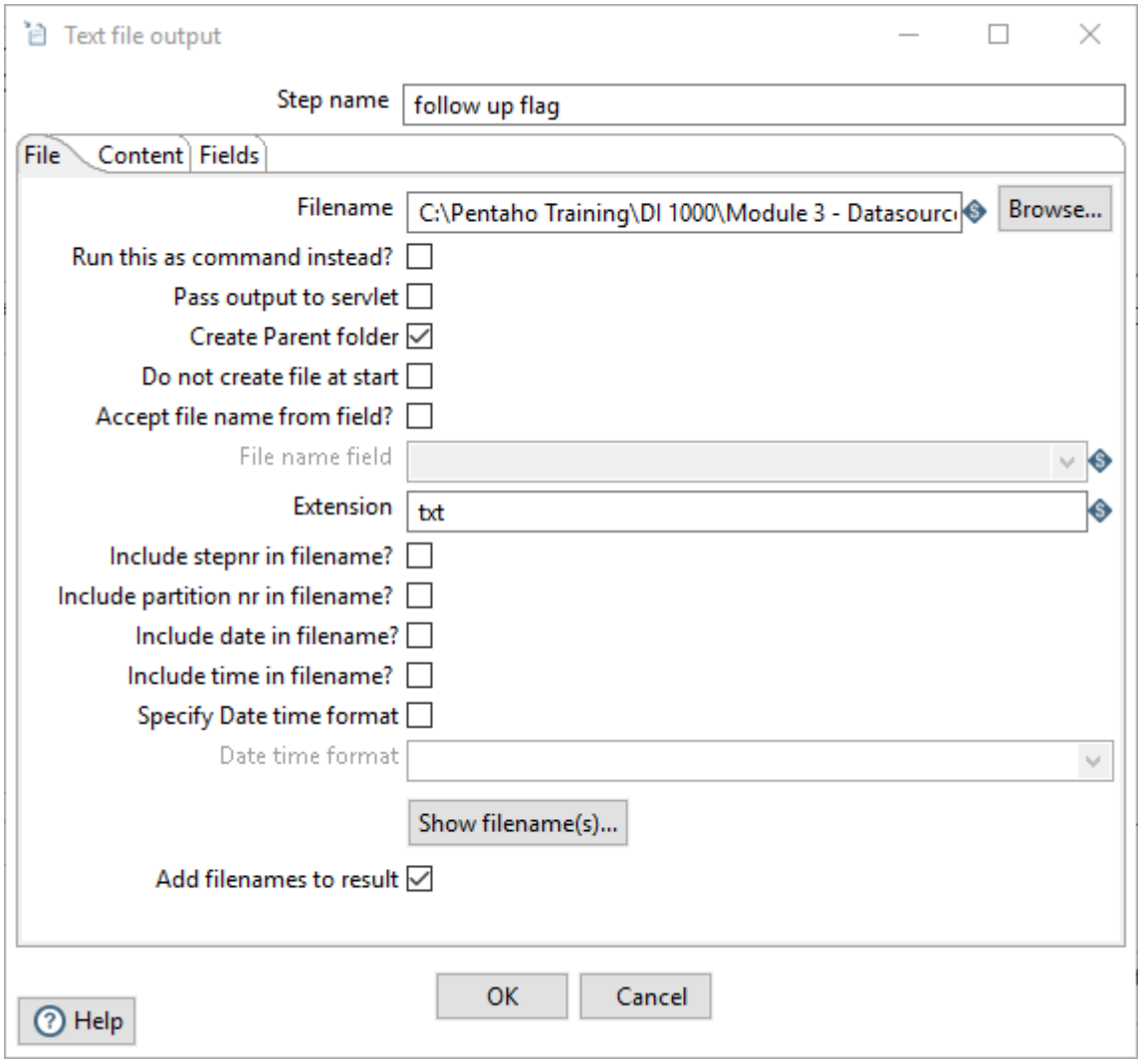
- This step 'splits' the data stream to Text File Output steps, based on the value associated with the action_flag stream field.

Text File Output - Follow Up

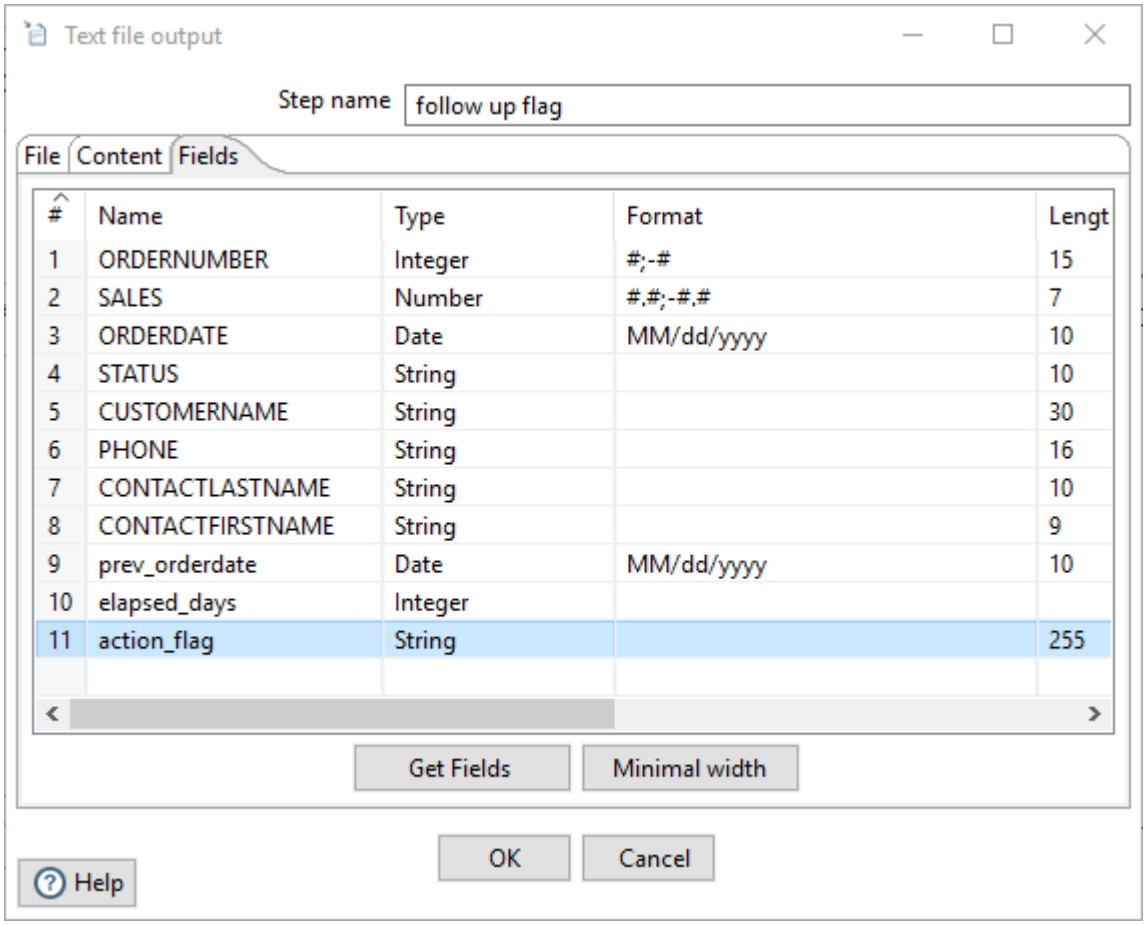
The Text file output step is used to export data to text file format. This is commonly used to generate Comma Separated Values (CSV files) that can be read by spreadsheet applications. It is also possible to generate fixed width files by setting lengths on the fields in the fields tab.

Note: It is not possible to execute this step on parallel to write to the same file. In this case, you need to set the option "Include stepnr in filename" and later merge the files.

1. Drag the Text File Output step onto the canvas.
2. Open the Text File Output properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click on the Fields tab, and ensure the following details are configured, as outlined below:



5. Click on the 'Get Fields' button, and remove non-essential fields. (action_flag)
6. Click OK.
7. Repeat the Text File Output steps to configure the other steps.

RUN the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Check the resulting text files.

```
ORDERNUMBER;ORDERDATE;CUSTOMERNAME;PHONE;CONTACTLASTNAME;CONTACTFIRSTNAME;sales_total;prev_orderdate;elapsed_days
10178;11/08/2003;Alpha Cognac                ;61.77.6555      ;Roulet    ;Annette  ;3492.5;07/04/2003; 127
10397;03/28/2005;Alpha Cognac                ;61.77.6555      ;Roulet    ;Annette  ;2577.6;11/08/2003; 506
10370;01/20/2005;Anna's Decorations, Ltd     ;02 9936 8555    ;O'Hara    ;Anna     ;2297.1;11/04/2003; 443
10298;09/27/2004;Atelier graphique           ;40.32.2555      ;Schmitt   ;Carine   ;3757.3;05/20/2003; 496
10265;07/02/2004;Australian Collectables, Ltd ;61-9-3844-6555 ;Connery   ;Sean     ;3907.8;11/21/2003; 224
10223;02/20/2004;Australian Collectors, Co.  ;03 9520 4555    ;Ferguson  ;Peter    ;3965.7;05/21/2003; 275
10342;11/24/2004;Australian Collectors, Co.  ;03 9520 4555    ;Ferguson  ;Peter    ;6454.4;02/20/2004; 278
10374;02/02/2005;Australian Gift Network, Co ;61-7-3844-6555 ;Calaghan  ;Tony     ;5288;11/06/2003; 454
10304;10/11/2004;Auto Associ,s & Cie.        ;30.59.8555      ;Tonini    ;Daniel   ;10172.7;02/02/2004; 252
10252;05/26/2004;Auto Canal+ Petit           ;(1) 47.55.6555  ;Perrier   ;Dominique;1527.8;01/15/2004; 132
10402;04/07/2005;Auto Canal+ Petit           ;(1) 47.55.6555  ;Perrier   ;Dominique;5833.8;05/26/2004; 316
10290;09/07/2004;Auto-Moto Classics Inc.     ;6175558428      ;Taylor    ;Leslie   ;2502;06/16/2003; 449
10306;10/14/2004;AV Stores, Co.              ;(171) 555-1555  ;Ashworth  ;Victoria ;6570.8;03/18/2003; 576
10158;10/10/2003;Baane Mini Imports          ;07-98 9555      ;Bergulfsen;Jonas    ;1474.7;01/29/2003; 254
10309;10/15/2004;Baane Mini Imports          ;07-98 9555      ;Bergulfsen;Jonas    ;4394.4;10/10/2003; 371
```

Introduction

The Update step first looks up a row in a table using one or more lookup keys. If it can be found and the fields to update are all the same, no action is taken, otherwise the record is updated.

Objectives

In this guided demonstration, you will:

- Update the Employees Table.
- Configure the following steps:
  - Update

Transformation

The stream fields are mapped to the database table by ensuring:
EMPLOYEE_NUMBER = EMPLOYEENUMBER

Ensure all the Field mappings are correct

Employee_update.txt

The Employee Table will be updated with the following records:

EMPLOYEE_NUMBER,LASTNAME,FIRSTNAME,EXT,EMAIL,OFFICE,REPORTS,TITLE

1002, Murphy, Diana,x5800,dmurphy@classicmodelcars.com,1,1000,CEO

1102, Bondur, Gerard,x5408,athompson@classicmodelcars.com,4,1056,Regional Sales Manager (EMEA)

Text File Input

The Text File Input step is used to read data from a variety of different text-file types. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

The Text File Input step provides you with the ability to specify a list of files to read, or a list of directories with wild cards in the form of regular expressions. In addition, you can accept filenames from a previous step making filename handling more even more generic.

1. Drag the Text File Input step onto the canvas.
2. Open the Text File properties dialog box.
3. Ensure the following details are configured, as outlined below:

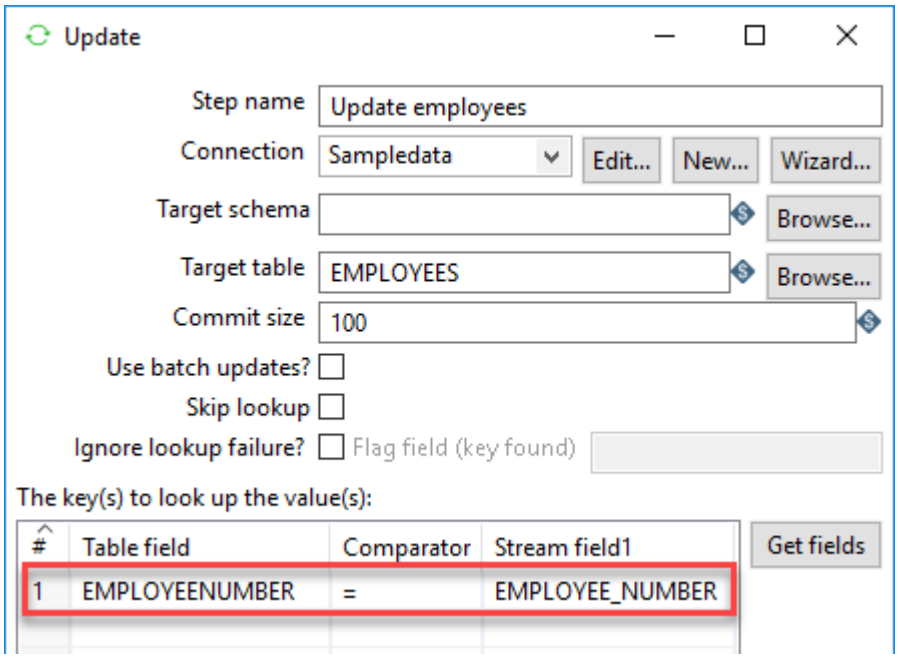
Stepname	Text File Input
Filename	[path to demo file]\employees_update.txt
Delimiter	Comma ,
Lazy conversion	Unchecked
Header row	Checked

4. Remember to Click 'Get Fields'.
5. Click OK.

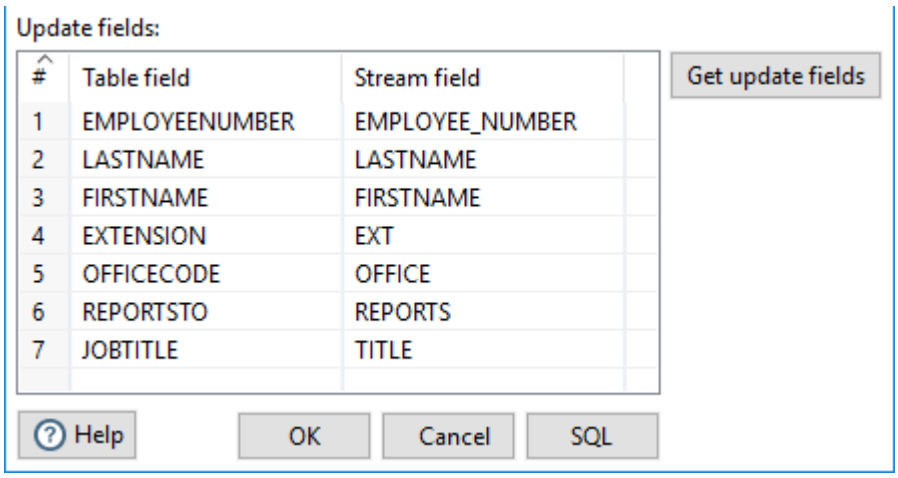
Update

The Update step first looks up a row in a table using one or more lookup keys. If the row matches the lookups, then it updates the record.

1. Drag the Update step onto the canvas.
2. Open the Update properties dialog box.
3. Ensure the following details are configured, as outlined below:



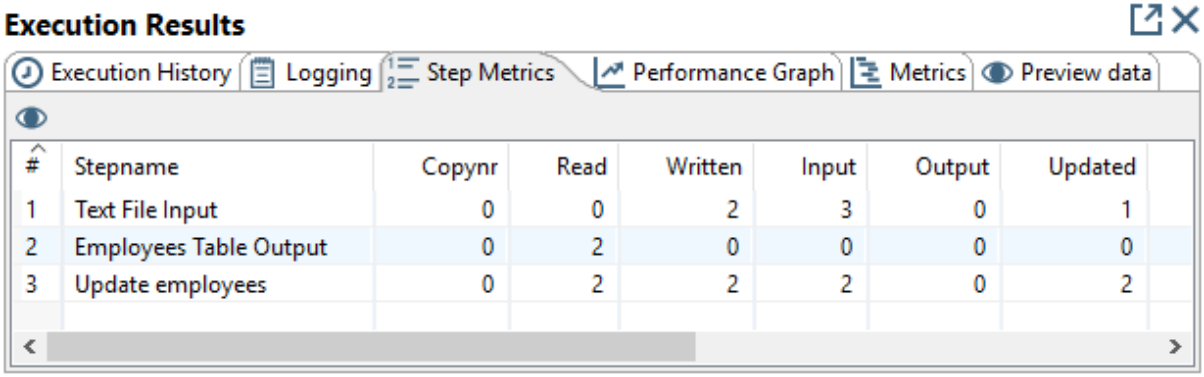
4. Next click the 'Get update fields' button to return the fields that are going to be updated.



5. Click Ok.

RUN the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Click on the Step Metrics tab:



#	Stepname	Copynr	Read	Written	Input	Output	Updated
1	Text File Input	0	0	2	3	0	1
2	Employees Table Output	0	2	0	0	0	0
3	Update employees	0	2	2	2	0	2

As expected, 2 records have been updated.

Guided Demo 3-2-5: Table Insert / Update

Introduction

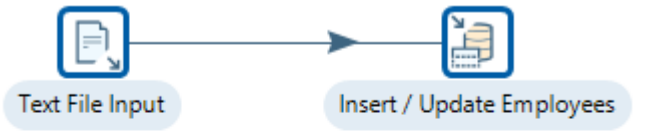
Steel Wheels needs to update its Employee records as folks join and get promoted.

Objectives

In this guided demonstration, you will:

- Insert / Update the Employees Table.
- Configure the following steps:
  - Insert / Update

Transformation



The stream fields are mapped to the database table by ensuring:
EMPLOYEE_NUMBER = EMPLOYEENUMBER

Ensure all the Field mappings are correct

Employee_insert_update.txt

The Employee Table will be updated / inserted with the following records:

EMPLOYEE_NUMBER, LASTNAME, FIRSTNAME, EXT, EMAIL, OFFICE, REPORTS, TITLE

1188,Firrelli,Julianne,x2174,jfirrelli@classicmodelcars.com,2,1143,Sales Manager

1619,King,Tom,x6324,tking@classicmodelcars.com,6,1088,Sales Rep

1810,Lundberg,Anna,x910,alundberg@classicmodelcars.com,2,1143,Sales Rep

1811,Schulz,Chris,x951,cschulz@classicmodelcars.com,2,1143,Sales Rep

Text File Input

The Text File Input step is used to read data from a variety of different text-file types. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

The Text File Input step provides you with the ability to specify a list of files to read, or a list of directories with wild cards in the form of regular expressions. In addition, you can accept filenames from a previous step making filename handling more even more generic.

1. Drag the Text File Input step onto the canvas.
2. Open the Text File properties dialog box.
3. Ensure the following details are configured, as outlined below:

Stepname	Text File Input
Filename	[path to demo file]\employees_insert_update.txt
Delimiter	Comma ,
Lazy conversion	Unchecked
Header row	Checked

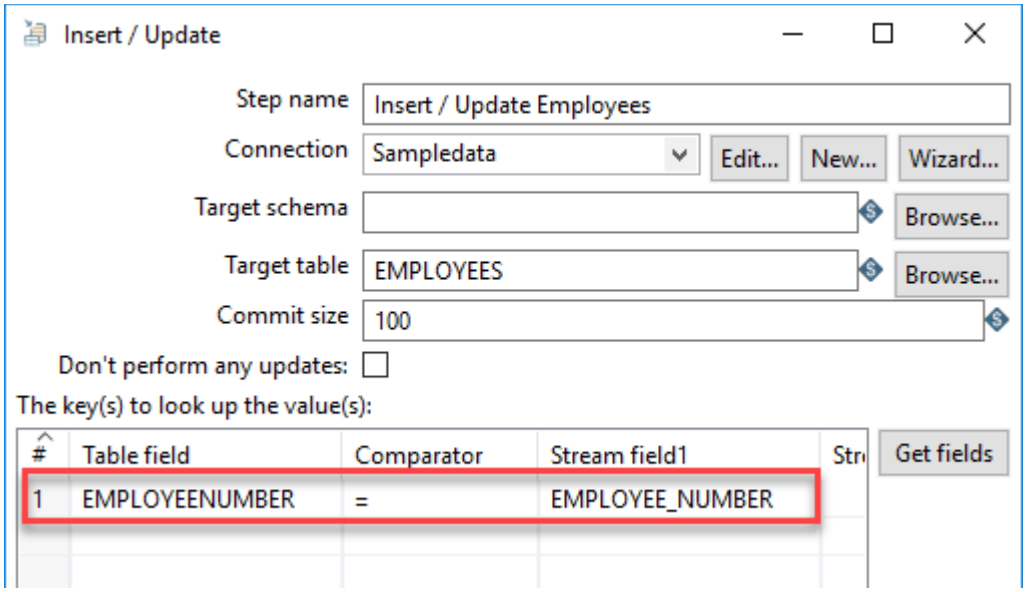
4. Remember to Click 'Get Fields'.
5. Click OK.

Insert / Update

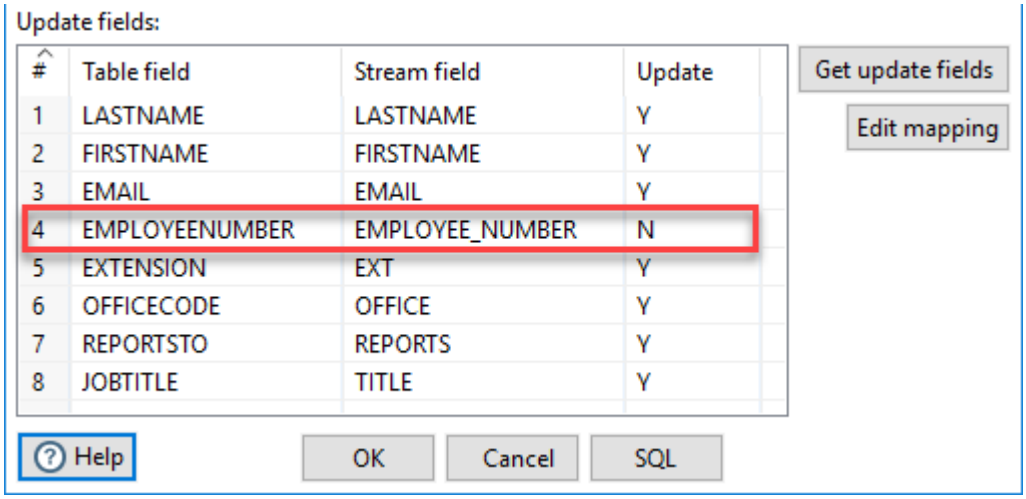
The Insert/Update step first looks up a row in a table using one or more lookup keys. If the row can't be found, it inserts the row. If it can be found and the fields to update are the same, nothing is done. If they are not all the same, the row in the table is updated.

Note: If you have multiple rows with the same keys that match, only the first row found is compared. This may lead to different results, depending on if the found row matches with given values or not. The update scenario looks like this: If a difference is found in the case of multiple rows with the same key, an UPDATE statement is fired against the database that updates all rows with the matching keys. This note also applies to the Update step

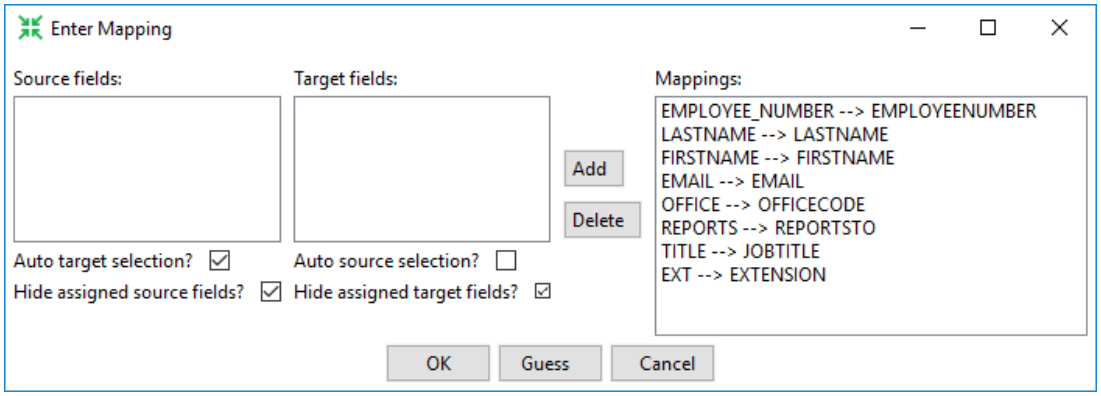
1. Drag the Insert / Update Input step onto the canvas.
2. Open the Insert / Update properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Next click the 'Get update fields' button to return the fields that are going to be updated.



5. Click on the 'Enter field mapping', and map accordingly:

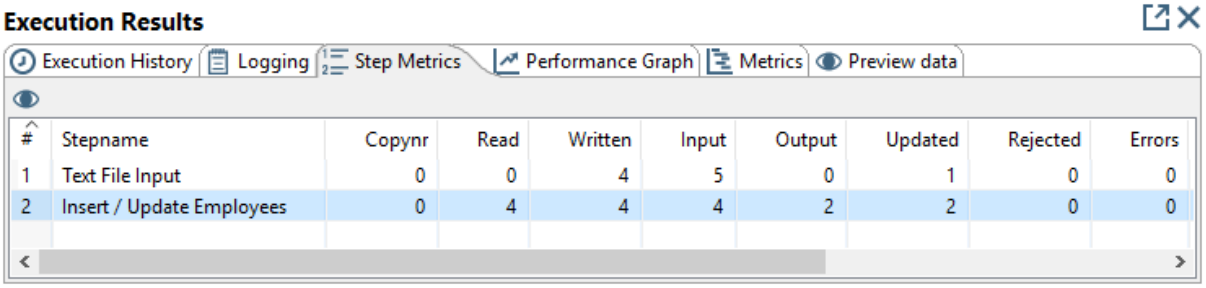


6. Click OK.

- This maps the stream fields to the database columns.

Run the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Click on the Step Metrics tab:



#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors
1	Text File Input	0	0	4	5	0	1	0	0
2	Insert / Update Employees	0	4	4	4	2	2	0	0

As expected 4 records have been written and 2 rows updated.

Slowly Changing Dimensions (SCD) - dimensions that change slowly over time, rather than changing on regular schedule, time-base. In a Data Warehouse, there is a need to track changes in dimension attributes in order to report historical data. In other words, implementing one of the SCD types should enable users assigning proper dimension's attribute value for given date. Example of such dimensions could be: customer, geography, employee.

There are many approaches how to deal with SCD. The most popular are:

Type 0 - The passive method

Type 1 - Overwriting the old value

Type 2 - Creating a new additional record

Type 3 - Adding a new column

Type 4 - Using historical table

Type 6 - Combine approaches of types 1,2,3 (1+2+3=6)

Type 0 - The passive method. In this method, no special action is performed upon dimensional changes. Some dimension data can remain the same as it was first time inserted, others may be overwritten.

Type 1 - Overwriting the old value. In this method, no history of dimension changes is kept in the database. The old dimension value is simply overwritten with the new one. This type is easy to maintain and is often use for data which changes are caused by processing corrections (e.g. removal special characters, correcting spelling errors).

Before the change:

Customer_ID	Customer_Name	Customer_Type
1	Cust_1	Corporate

After the change:

Customer_ID	Customer_Name	Customer_Type
1	Cust_1	Retail

Type 2 - Creating a new additional record. In this methodology, all history of dimension changes is kept in the database. You capture attribute change by adding a new row with a new surrogate key to the dimension table. Both the prior and new rows contain as attributes the natural key (or another durable identifier). Also 'effective date' and 'current indicator' columns are used in this method. There could be only one record with current indicator set to 'Y'. For 'effective date' columns, i.e. start_date and end_date, the end_date for current record usually is set to value 9999-12-31. Introducing changes to the dimensional model in type 2 could be very expensive database operation so it is not recommended to use it in dimensions where a new attribute could be added in the future.

Before the change:

Customer_ID	Customer_Name	Customer_Type	Start_Date	End_Date	Version
1	Cust_1	Corporate	22-07-2010	31-12-9999	1

After the change:

Technical Key	Customer_ID	Customer_Name	Customer_Type	Start_Date	End_Date	Version
1	1	Cust_1	Corporate	22-07-2010	17-05-2012	1
2	1	Cust_1	Retail	17-05-2012	31-12-9999	2

Type 3 - Adding a new column. In this type, usually only the current and previous value of dimension is kept in the database. The new value is loaded into 'current/new' column and the old one into 'old/previous' column. The history is limited to the number of column created for storing historical data. This is the least commonly needed technique.

Before the change:

Customer_ID	Customer_Name	Current_Type	Previous_Type
1	Cust_1	Corporate	Corporate

After the change:

Customer_ID	Customer_Name	Current_Type	Previous_Type
1	Cust_1	Retail	Corporate

Type 4 - Using historical table. In this method, a separate historical table is used to track all dimension's attribute historical changes for each of the dimension. The 'main' dimension table keeps only the current data e.g. customer and customer_history tables.

Current table:

Customer_ID	Customer_Name	Customer_Type
1	Cust_1	Corporate

Historical table:

Customer_ID	Customer_Name	Customer_Type	Start_Date	End_Date
1	Cust_1	Retail	01-01-2010	21-07-2010
1	Cust_1	Oher	22-07-2010	17-05-2012
1	Cust_1	Corporate	18-05-2012	31-12-9999

Type 6 - Combine approaches of types 1,2,3 (1+2+3=6). In this type, we have in dimension table such additional columns as:

current_type - for keeping current value of the attribute. All history records for given item of attribute have the same current value.

historical_type - for keeping historical value of the attribute. All history records for given item of attribute could have different values.

start_date - for keeping start date of 'effective date' of attribute's history.

end_date - for keeping end date of 'effective date' of attribute's history.

current_flag - for keeping information about the most recent record.
In this method to capture attribute change we add a new record as in type 2. The current_type information is overwritten with the new one as in type 1. We store the history in a historical_column as in type 3.

Customer_ID	Customer_Name	Current_Type	Historical_Type	Start_Date	End_Date	Current_Flag
1	Cust_1	Corporate	Retail	01-01-2010	21-07-2010	N
2	Cust_1	Corporate	Other	22-07-2010	17-05-2012	N
3	Cust_1	Corporate	Corporate	18-05-2012	31-12-9999	Y

Introduction

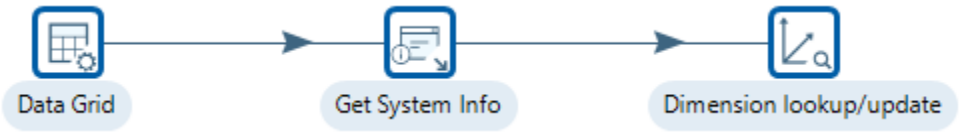
The demonstration illustrates the key concepts and workflows for dealing with Slowly Changing Dimensions.

Objectives

In this guided demonstration, you will:

- Learn about the Step modes:
  - Dimension Update
  - Dimension Lookup
- Missing Natural Key
- Configure the following steps:
  - Dimension Lookup / Update

Transformation



In this demonstration, you are going to create a dimension table DIM_SCD, which will have:

- a technical key field
- a version field
- a date from and date to fields
- an Id field
- a name field
- The technical key will become the primary key for the dimension table.
- The version field sets the record version (increments) on an update or insert.

Create the DIM_SCD table

1. In your SQL Query Tool, execute the following statement:

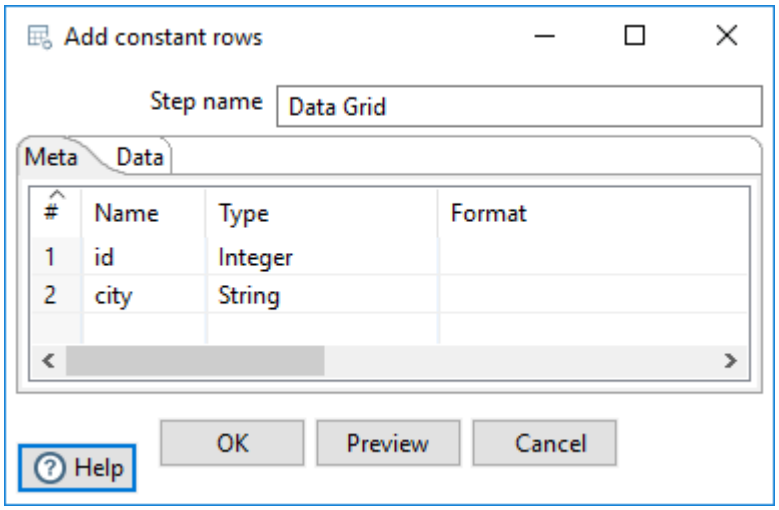
```
CREATE TABLE PUBLIC.DIM_SCD (
TK BIGINT NOT NULL,
VERSION INTEGER,
DATE_FROM TIMESTAMP,
DATE_TO TIMESTAMP,
ID INTEGER,
CITY VARCHAR (20),
LAST_UPDATE TIMESTAMP,
PRIMARY KEY (TK)
);
```

- This prepares the table for Workflow 2: SCD Type II with LAST_UPDATE field
- The script is in the GD 3-2-6 folder

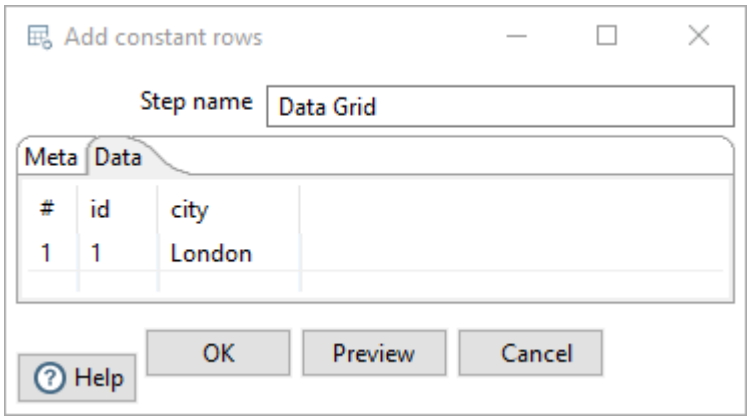
Data Grid

This step allows you to enter a static list of rows in a grid. This is usually done for testing, reference or demo purposes.

1. Drag the Data Grid step onto the canvas.
2. Open the Data Grid properties dialog box.
3. Ensure the following details are configured, as outlined below:
4. On the Meta tab enter the following values:



5. Click on the Data tab and add the following values:

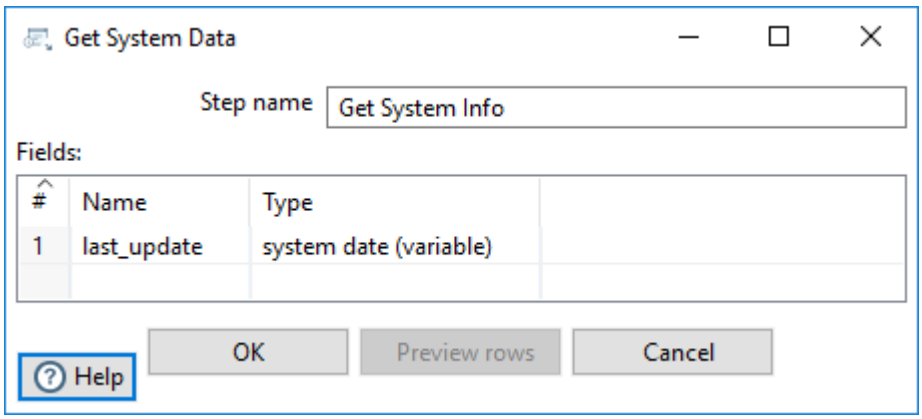


Get System Info

The Get System Info step retrieves information from the Kettle environment. The table below contains the available information types.

This step generates a single row with the fields containing the requested information. It also accepts input rows. The selected values are added to the rows found in the input stream(s).

1. Drag the Get System Info step onto the canvas.
2. Open the Get System Info properties dialog box.
3. Ensure the following details are configured, as outlined below:



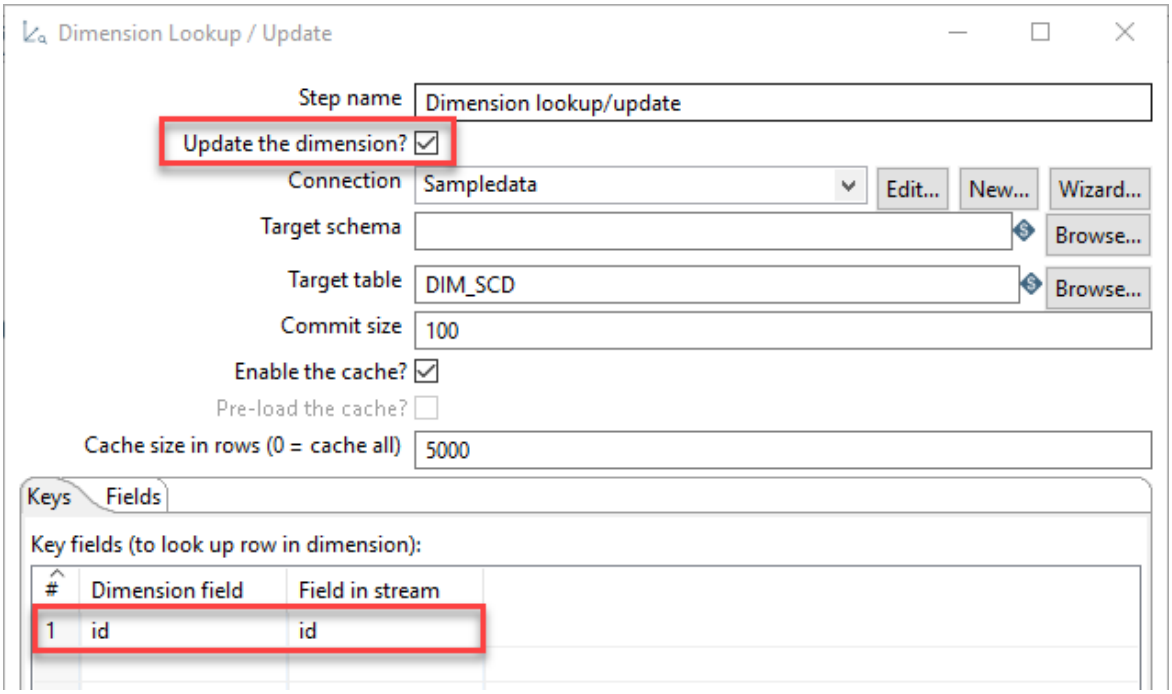
Workflow 1

Dimension Insert Mode

The Dimension Lookup/Update step allows you to implement Ralph Kimball's slowly changing dimension for both types: Type I (update) and Type II (insert) together with some additional functions.

Not only can you use this step to update a dimension table, it may also be used to look up values in a dimension.

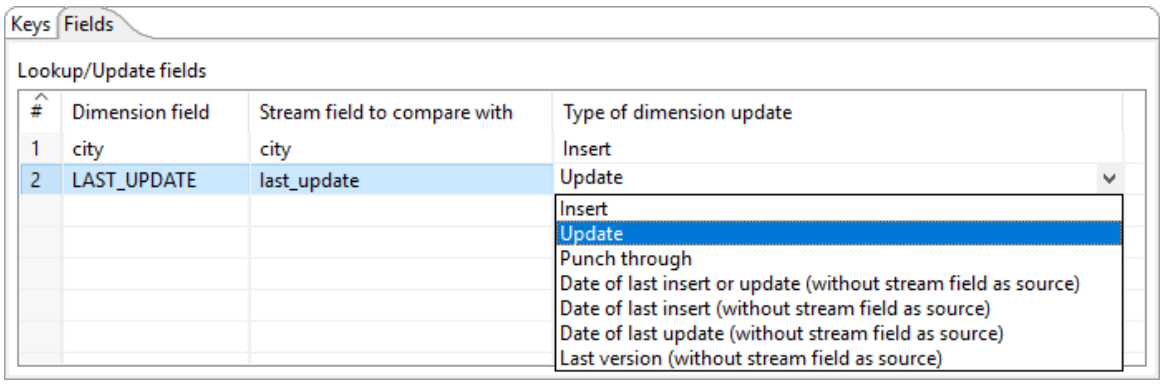
1. Drag the Dimension Lookup / Update step onto the canvas.
2. Open the Dimension Lookup / Update properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click on the Get Fields button. This will add the key fields used in the Lookup.

- The keys that map the records (rows) in the dimension table to the stream field is: id

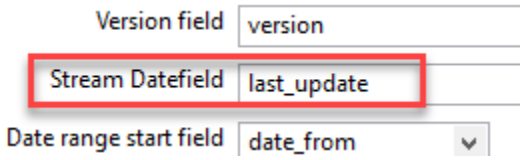
5. Click on the Fields tab and map the stream name field to the dimension name field and ensure type of dimension update is insert.
6. Click on the Get Fields button. This will add the stream fields.

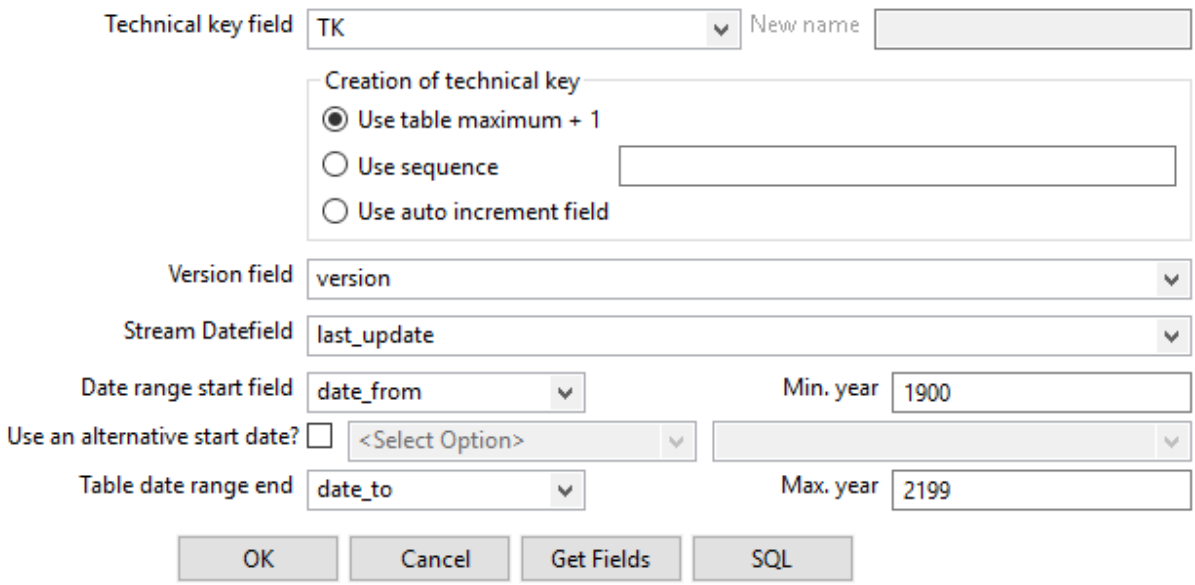


There are several options available to insert / update the dimension record (row).

- Insert: This option implements a Type II slowly changing dimension policy. If the difference is detected for one or more mappings that have the Insert option, then a row is added to the dimension table.
- Update: This option simply updates the matched row. It can be used to implement a Type I slowly changing dimension.
- Punch through: The punch through option also performs an update. But instead of only updating the matched dimension row, it will update all versions of the row in a Type II slowly changing dimension.
- Date of last insert or update (without stream field as source): Use this option to let the step automatically maintain a date field that records the date of the insert or update using the system date field as source.
- Date of last insert (without stream field as source): Use this option to let the step automatically maintain a date field that records the date of the last insert using the system date field as source.
- Date of last update (without stream field as source): Use this option to let the step automatically maintain a date field that records the date of the last update using the system date field as source.
- Last version (without stream field as source): Use this option to let the step automatically maintain a flag that indicates if the row is the last version.

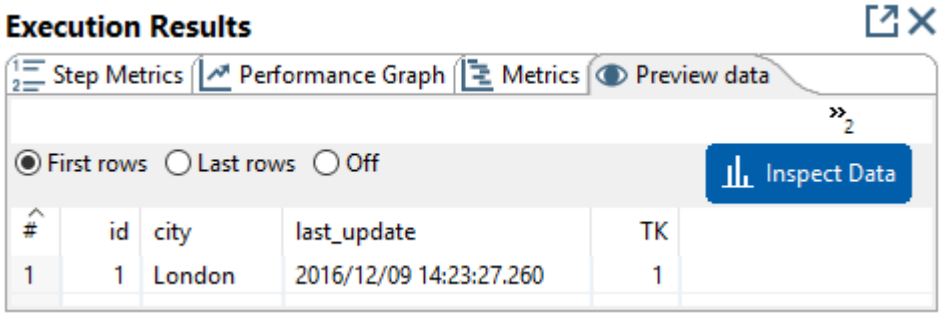
1. The Stream Datefield can now be set: last_update.





RUN the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Click on the Preview tab:



3. Examine the table in your SQL Query Tool

	TK	VERSION	DATE_FROM	DATE_TO	ID	CITY	LAST_UPDATE
1	0	1					
2	1	1	01/01/1900 00:00:00.000	12/31/2199 23:59:59.999	1	London	12/09/2016 14:23:27.260

- Kettle automatically inserts an additional record with a technical key of value 0 (for default or unknown values). This will only happen in the first execution. Below this record, you find the one record (London) from our sample dataset.

- In update mode (update option is enabled) the step first performs a lookup of the dimension entry. The result of the lookup is different though. Not only the technical key is retrieved from the query, but also the dimension attribute fields. A field-by-field comparison then follows. The result can be one of the following situations:
    - The record was not found, we insert a new record in the table.
    - The record was found and any of the following is true:

      One or more attributes were different and had an "Insert" (Kimball Type II) setting: A new dimension record version is inserted.

      One or more attributes were different and had a "Punch through" (Kimball Type I) setting: These attributes in all the dimension record versions are updated.

      One or more attributes were different and had an "Update" setting: These attributes in the last dimension record version are updated.

      All the attributes (fields) were identical: No updates or insertions are performed.

  Note: If you mix Insert, Punch Through and Update options in this step, this algorithm acts like a Hybrid Slowly Changing Dimension. (it is no longer just Type I or II, it is a combination)

Workflow 2

Dimension Insert / Update Type II Mode

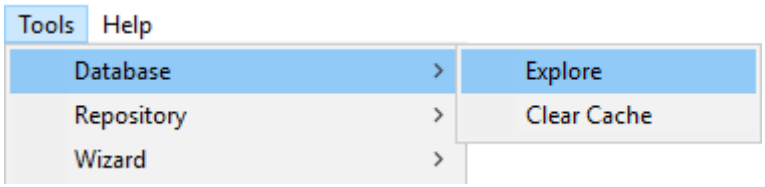
Let's truncate the DIM_SCD table..!

1. From your SQL Query Tool, execute the following statement:

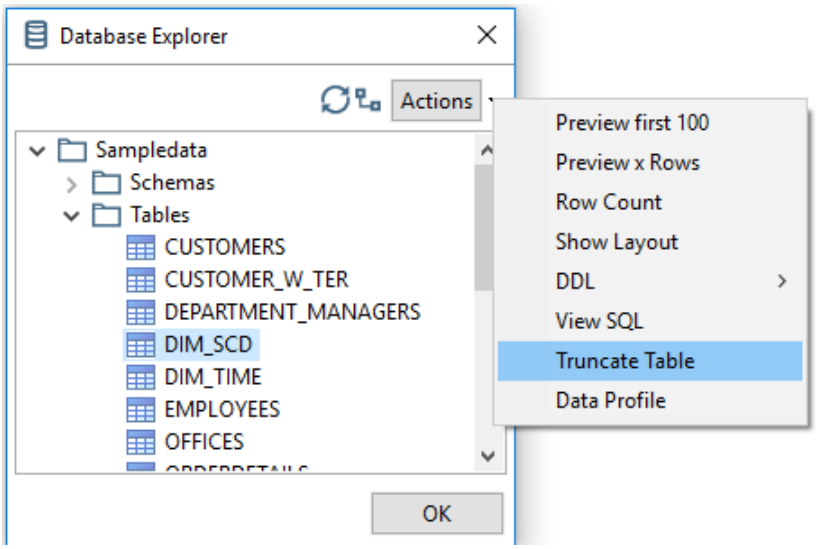
   ```
   truncate table DIM_SCD
   ```

   Or

2. From the main menu, select: Tools > Database > Explore

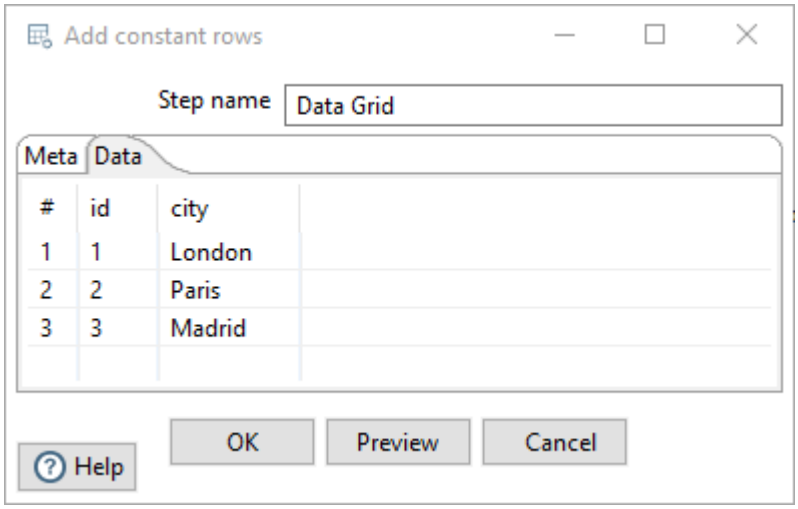


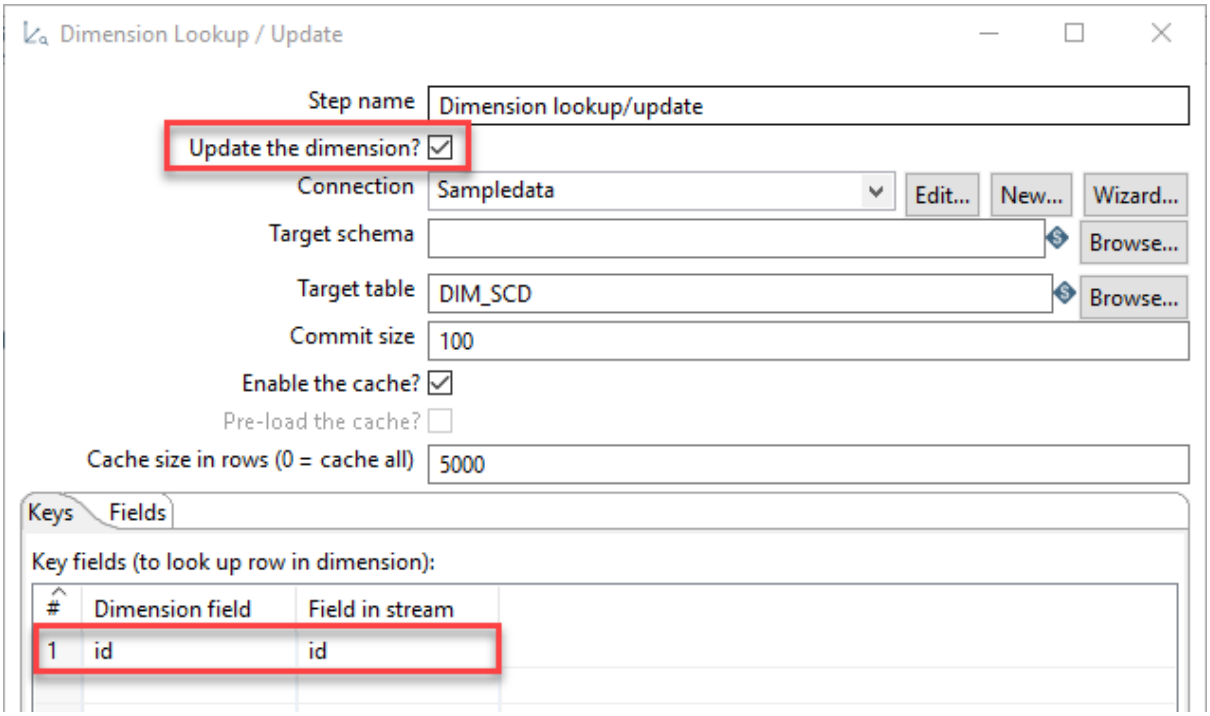
3. Select the DIM_SCD table



4. From the Actions, drop-down box, select Truncate table

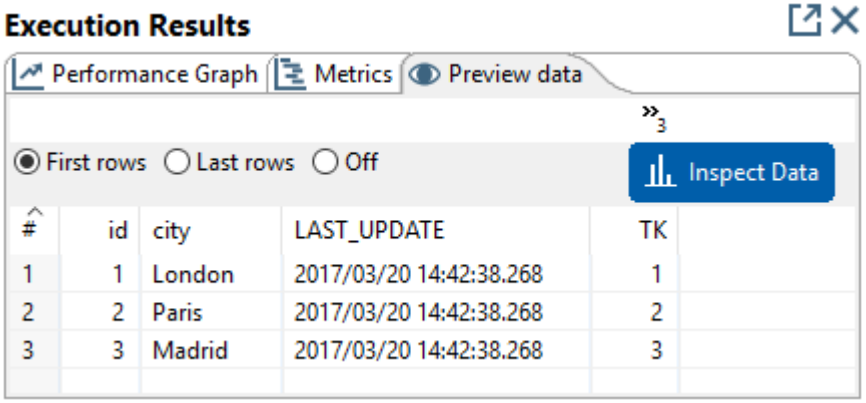
5. Modify the Data Grid values as illustrated:





Run the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Examine the table in your SQL Query Tool:

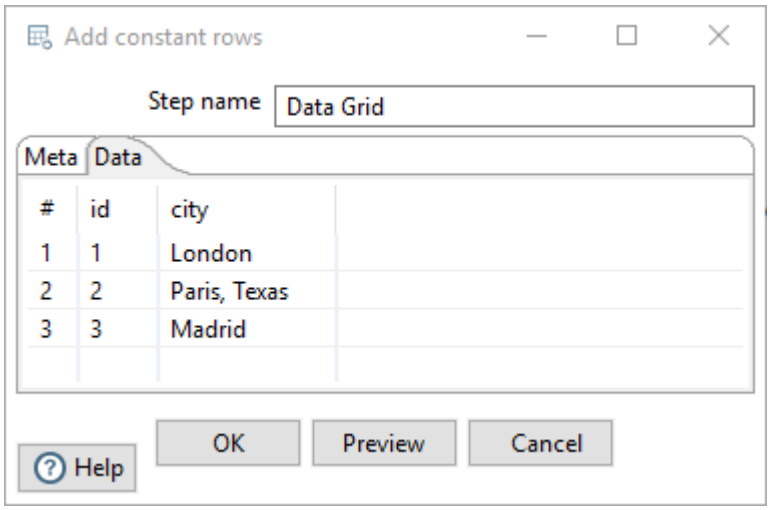


TK	VERSION	DATE_FROM	DATE_TO	ID	CITY	LAST_UPDATE
0	1					
1	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	1	London	2017-03-20 14:42:38.268000
2	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	2	Paris	2017-03-20 14:42:38.268000
3	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	3	Madrid	2017-03-20 14:42:38.268000

- As expected 3 records inserted

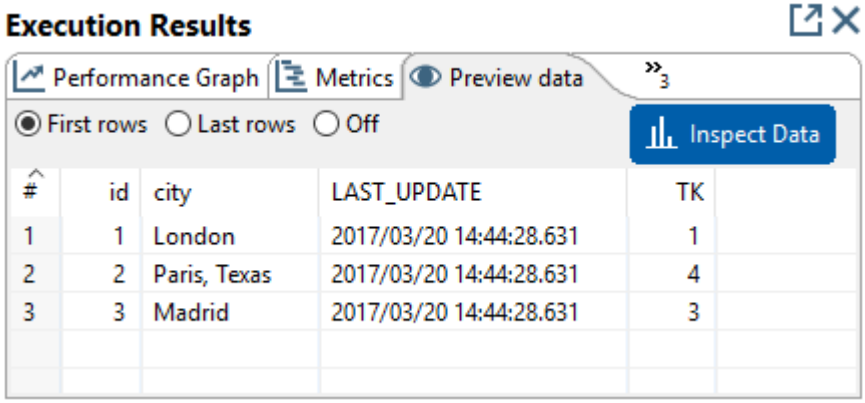
Let's now edited one of the values:

1. Modify the Data Grid step; add the 'Texas' value to Paris.



Run the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Examine the table in your SQL Query Tool:



	TK	VERSION	DATE_FROM	DATE_TO	ID	CITY	LAST_UPDATE
1	0	1					
2	1	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	1	London	2017-03-20 14:44:28.631000
3	2	1	1900-01-01 00:00:00.000000	2017-03-20 14:44:28.423000	2	Paris	2017-03-20 14:42:38.268000
4	3	1	1900-01-01 00:00:00.000000	2199-12-31 23:59:59.999000	3	Madrid	2017-03-20 14:44:28.631000
5	4	2	2017-03-20 14:44:28.423000	2199-12-31 23:59:59.999000	2	Paris, T...	2017-03-20 14:44:28.631000

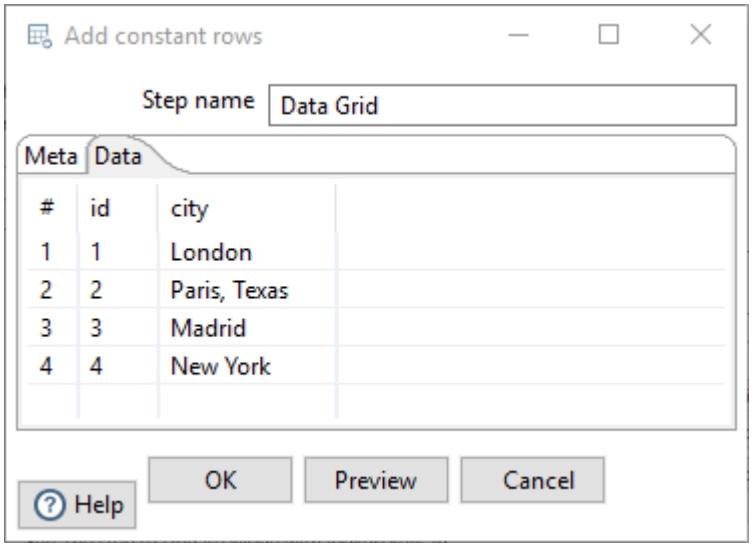
- A new record has been inserted, with the value 'Paris, Texas', with an updated date_from and last_update timestamp, and version.
- Notice that the LAST_UPDATE field has also been updated for the other records.

If you have time, try out various combinations of 'Type of Dimension update'.

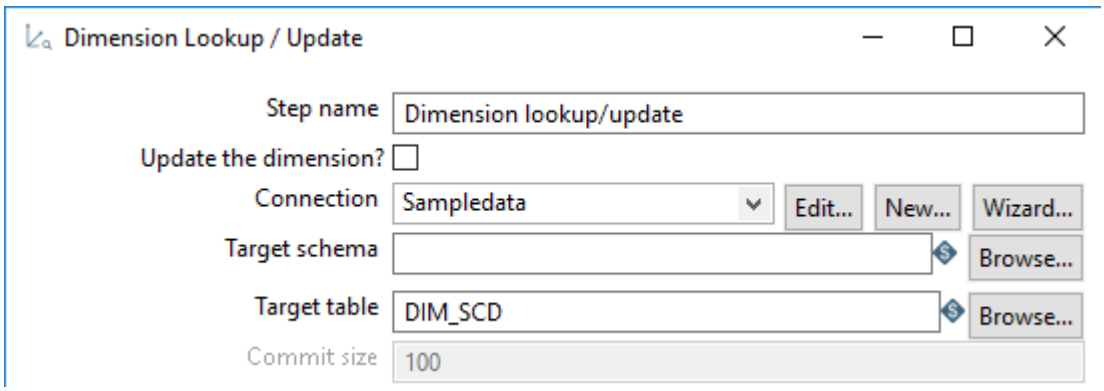
Workflow3

Dimension Lookup Mode

1. Add 'New York' to the Data Grid.

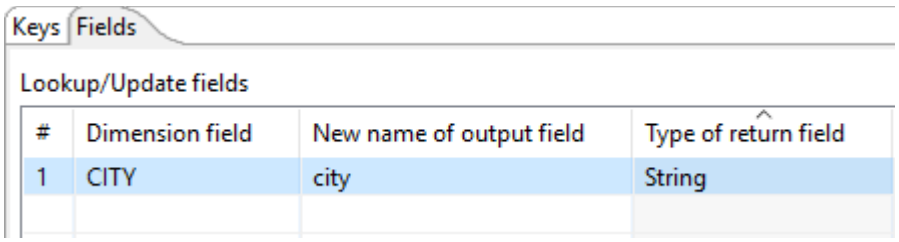


2. Ensure that the Dimension Lookup / Update is set to: Lookup Mode.



- Sets the step to Update Mode with lookup keys:id

2. Click on the Fields tab:

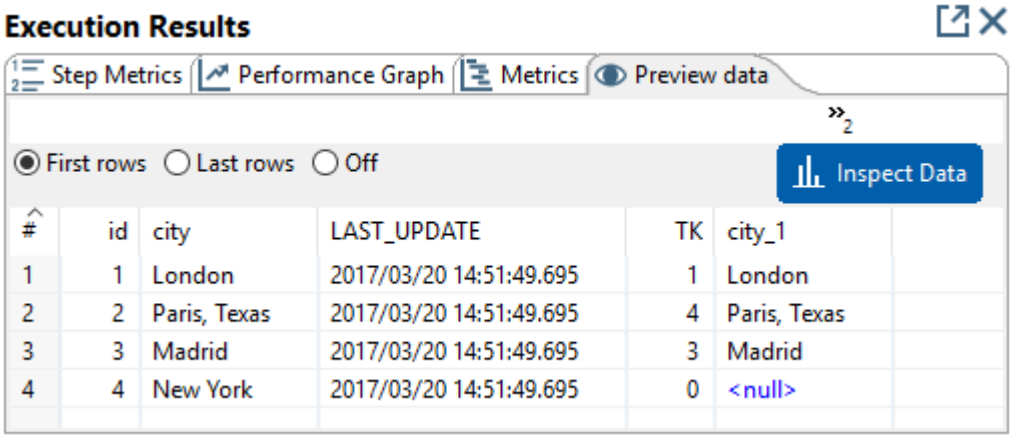


- Notice the change in column field names

3. Remove the LAST_UPDATE field. There's no point returning th last_update field as we're just after the values for CITY.

Run the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Examine the table in your SQL Query Tool:



#	id	city	LAST_UPDATE	TK	city_1
1	1	London	2017/03/20 14:51:49.695	1	London
2	2	Paris, Texas	2017/03/20 14:51:49.695	4	Paris, Texas
3	3	Madrid	2017/03/20 14:51:49.695	3	Madrid
4	4	New York	2017/03/20 14:51:49.695	0	<null>

- Why does New York have a TK 0 and a value of NULL?
- To maintain referential integrity, the record is assigned a TK 0, and as it doesn't exist in the database, the value returned is NULL. As this is a lookup no records are written to the database table.

If you have time see what happens when no natural key exists..

Deleting Columns

Sometimes you might have to delete data from a table. If the operation to do it is simple, for example:

```
DELETE FROM ORDERS_TABLE WHERE STATUS='Shipped'
```

Or

```
DELETE FROM TMP_TABLE
```

You could simply execute it by using an SQL job entry or an Execute SQL script step. If you face the second of the above situations, you can even use a Truncate table job entry.

For more complex scenarios, you should use the Delete step.

Guided Demo 3-2-7: Delete Records

Introduction

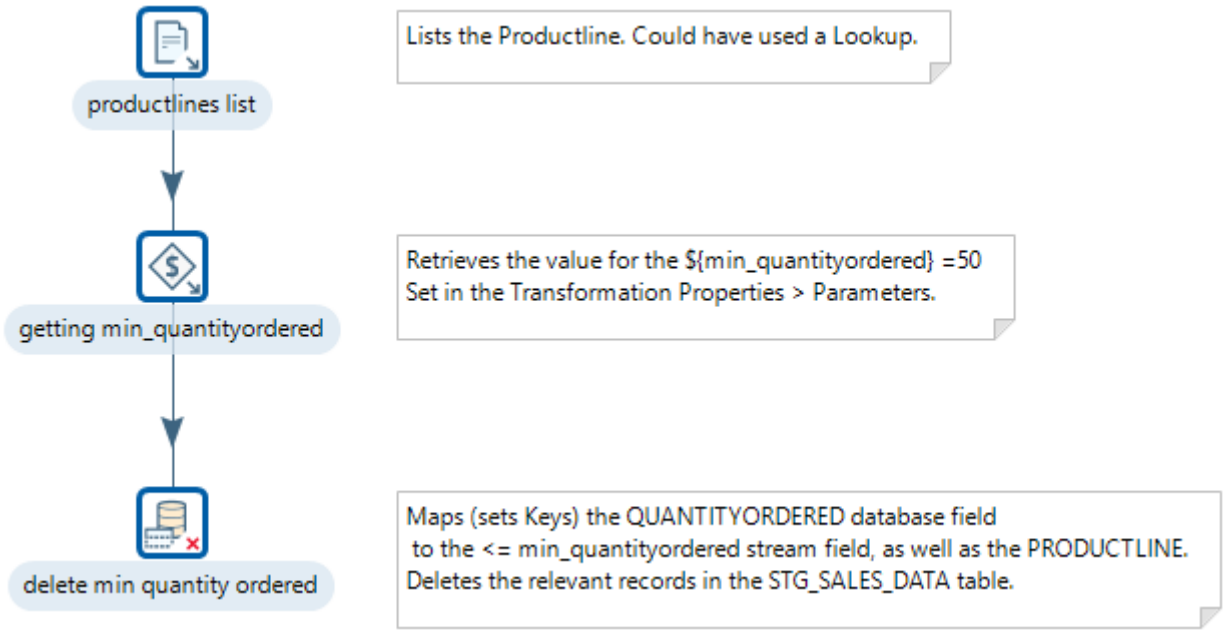
Steel Wheels are launching a campaign, focusing on Customers who have ordered more than 50 of each of their various Productlines.

Objectives

In this guided demonstration, you will:

- Configure the following steps:
  - Get variable
  - Lookup Up Database table
  - Delete

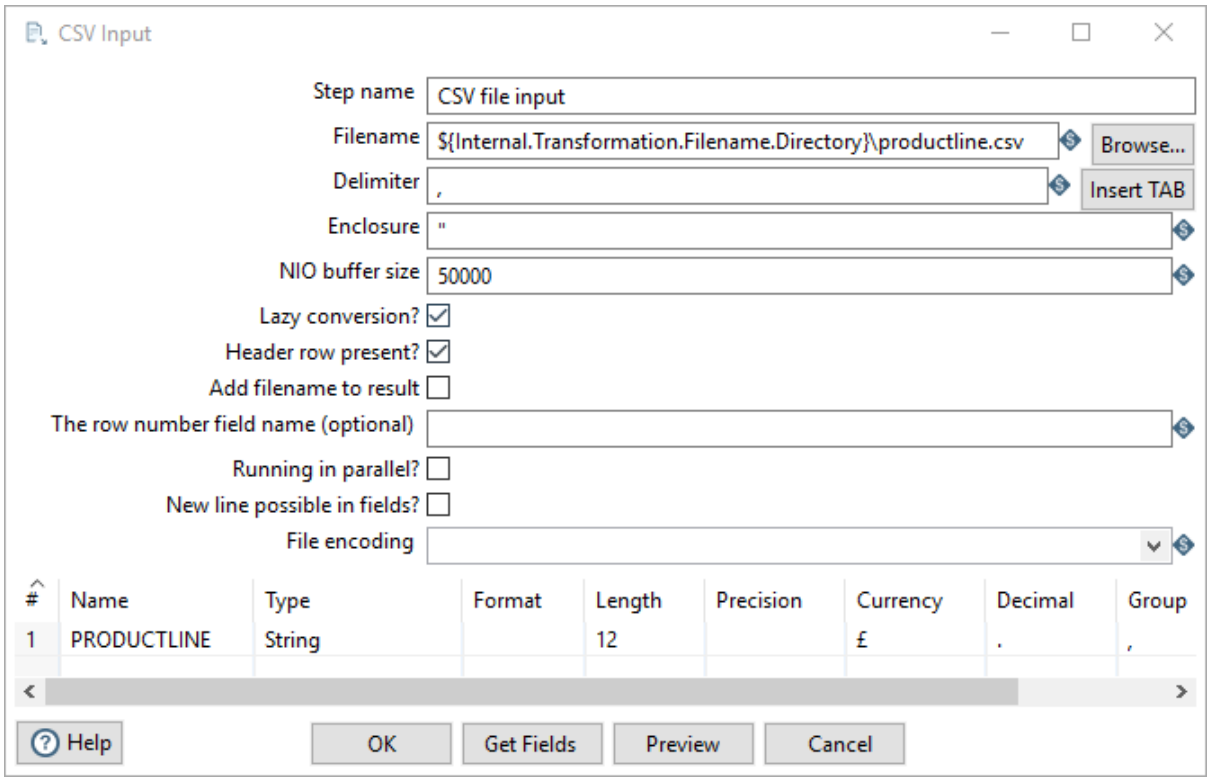
Transformation



CSV File Input

This step provides the ability to read data from a delimited file. The CSV label for this step is a misnomer because you can define whatever separator you want to use, such as pipes, tabs, and semicolons; you are not constrained to using commas. Internal processing allows this step to process data quickly. Options for this step are a subset of the Text File Input step.

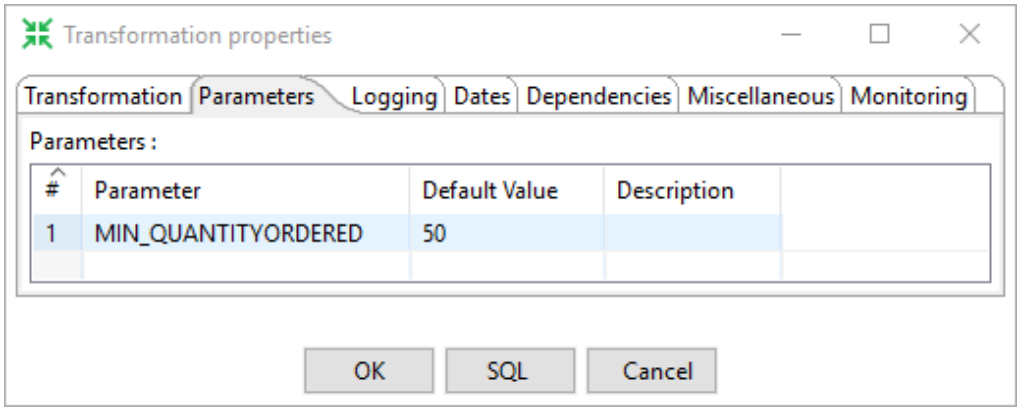
1. Drag the CSV File Input step onto the canvas.
2. Open the CSV File Input properties dialog box.
3. Ensure the following details are configured, as outlined below:



- List the Productline values.

Set ${MIN_QUANTITYORDERED} Parameter

1. Double-click on the canvas and select the Parameter tab.
2. Ensure the following details are configured, as outlined below:



Get variables

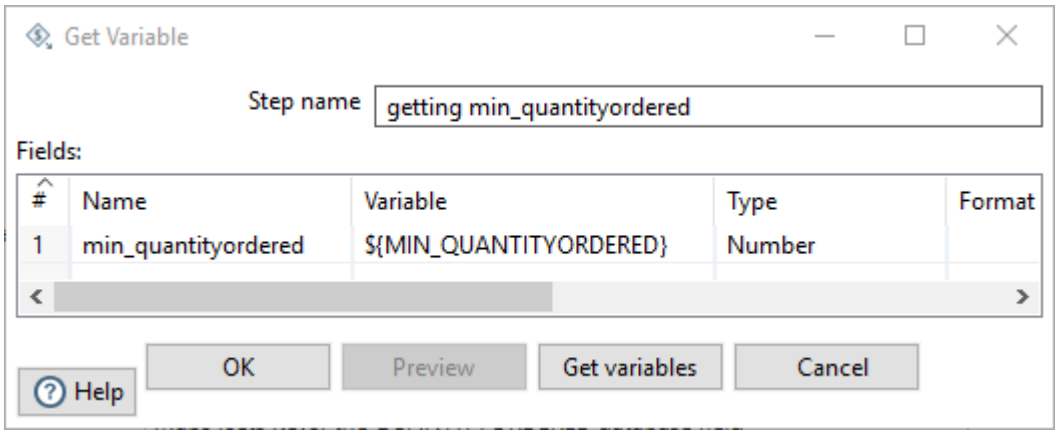
This step allows you to get the value of a variable. This step can return rows or add values to input rows.

Note: You must specify the complete variable specification in the format ${variable} or %%variable%% (as described in Variables). That means you can also enter complete strings in the variable column, not just a variable.

For example, you can specify: ${java.io.tmpdir}/kettle/tempfile.txt and it will be expanded to /tmp/kettle/tempfile.txt on Unix-like systems.

To convert the Variable into a data type other than String use Select Values - Meta Data tab.
To get system values, including command line arguments, use the Get System Info step.

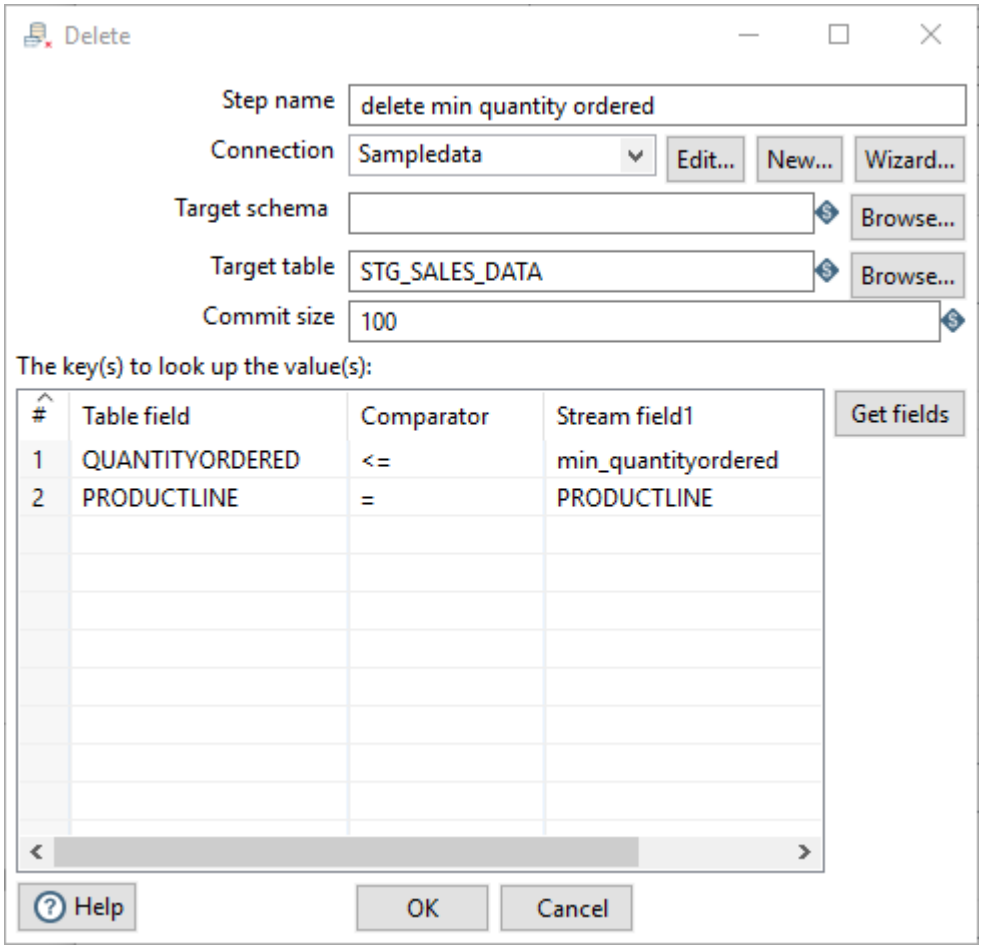
1. Drag the Get variables step onto the canvas.
2. Open the Get variables properties dialog box.
3. Ensure the following details are configured, as outlined below:



Delete

Operates in the same way as Truncate. Deletes rows.

1. Drag the Delete step onto the canvas.
2. Open the Delete properties dialog box.
3. Ensure the following details are configured, as outlined below:



- The value of the QUANTITYORDERED is set: greater / equal to the min_quantityordered.
- PRODUCTLINE values mapped.
- Delete the records from the STG_SALES_DATA table.

RUN the Transformation

1. Run the Transformation.
2. Compare the 'QUANTITYORDERED' records between the SALES_DATA and STG_SALES_DATA.

	ORDERNUMBER	QUANTITYORDERED
1	10107	30
2	10121	34
3	10134	41
4	10145	45
5	10168	36
6	10180	29
7	10188	48
8	10211	41
9	10223	37
10	10237	23
11	10251	28
12	10263	34
13	10275	45
14	10285	36
15	10299	23
16	10309	41

SALES_DATA

	ORDERNUMBER	QUANTITYORDERED
1	10417	66
2	10403	66
3	10417	56
4	10400	64
5	10341	55
6	10403	66
7	10412	54
8	10341	55
9	10405	97
10	10424	54
11	10406	61
12	10422	51
13	10407	59
14	10407	76
15	10342	55
16	10405	61

STG_SALES_DATA

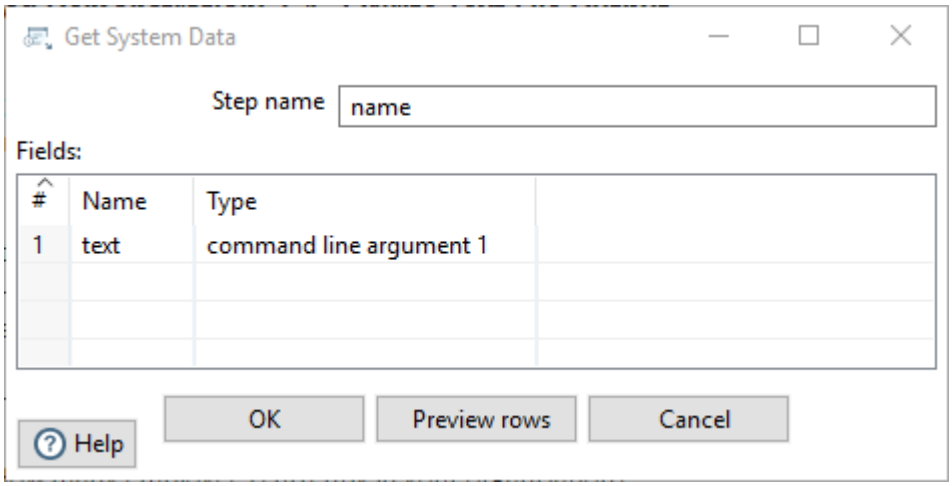
When you Run a Transformation or Job, there are several ways you can define the required settings.

- Arguments are used to send command line arguments to the Transformation.
- Named parameters are a system that allows you to parameterize your transformations and jobs.
- Variables allow you to dynamically enter the required setting.

Arguments

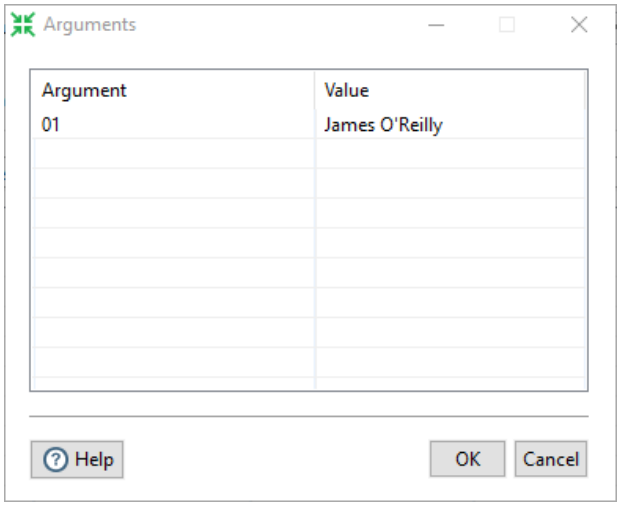
If you just need the arguments for one step only, then you can use the Get System Info step and create a hop to your Input step.

This approach was taken in GD3-1-2: Write Text Output, where the Customer name is populated with a Get System Info step.



To set the Argument:

1. Click on F8/F9, then the 'Arguments (Legacy) button.
2. The following value has been set for Argument 01



If you plan to use command line arguments in more than one Transformation or Step, then you should set the Arguments in a separate Transformation.

The Set Variables transformation has two steps:

- **Get System Info**: It allows you to define variables that are expected to come from the command line
- **Set Variables**: This one will then set the variables for the execution within Kettle, so that you can use them in the next transformation that is specified in your job.

The command line arguments enter Kettle as a String. In some cases, the variable is expected to be of a certain data type. Then you should use the Get Variable step in a preceding transformation to define the specific data type for each variable.

Parameters

Named parameters are special in the sense that they are explicitly named command line arguments. If you pass on a lot of arguments to your Kettle job or transformation, it might help to assign those values to an explicitly named parameter.

Named Parameters have following advantages:

- On the command line you assign the value directly to a parameter, hence there is zero chance of a mix-up.
- A default value can be defined for a named parameter
- A description can be provided for a named parameter
- No need for an additional transformation that sets the variables for the job

Variables

Variables can be used throughout Pentaho Data Integration, including in transformation steps and job entries. You define variables by setting them with the Set Variable step in a transformation or by setting them in the kettle.properties file in the directory:

$HOME/.kettle (Unix/Linux/OSX)

C:\Documents and Settings\<username>\.kettle\ (Windows XP)

C:\Users\<username>\.kettle\ (Windows Vista, 7 and later)

The way to use them is either by grabbing them using the Get Variable step or by specifying meta-data strings like:

${VARIABLE}

or:

%%VARIABLE%%

Both formats can be used and even mixed, the first is a UNIX derivative, the second is derived from Microsoft Windows. Dialogs that support variable usage throughout Pentaho Data Integration are visually indicated using a red dollar sign. You can use <CTRL>+ space hot key to select a variable to be inserted into the property value. Mouse over the variable icon to display the shortcut help.

Other ways to set and access variables:

- There are also System parameters, including command line arguments. These can be accessed using the Get System Info step in a transformation.
- You can also specify values for variables in the "Execute a transformation/job" dialog in Spoon or the Scheduling perspective. If you include the variable names in your transformation they will show up in these dialogs.
- It is also possible to set variables by Named Parameters.

Guided Demo 3-2-8: Parameters and Variables

Introduction

Named parameters are special in the sense that they are explicitly named command line arguments. If you pass on a lot of arguments to your Kettle job or transformation, it might help to assign those values to an explicitly named parameter.

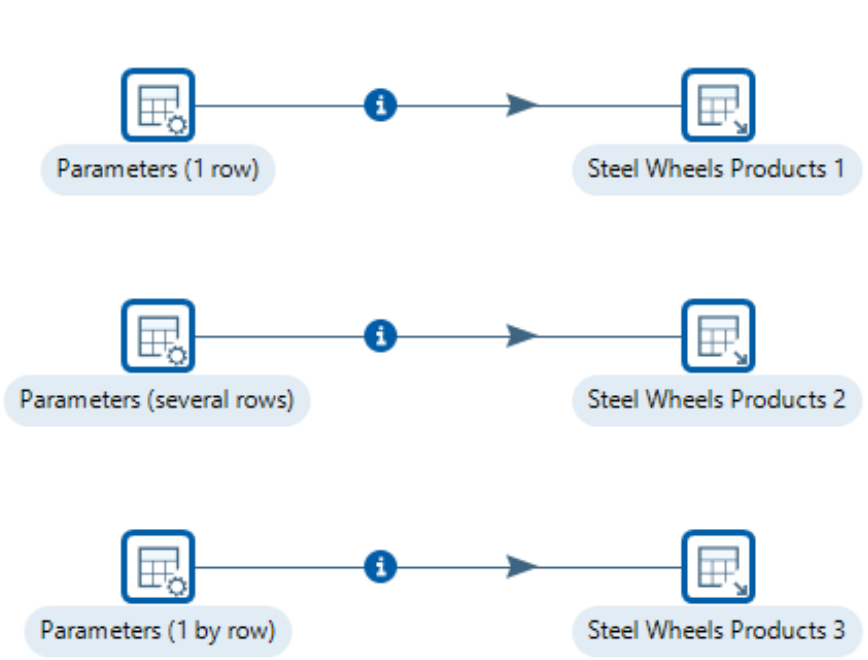
Objectives

In this guided demonstration, you will:

- Pass various Parameters by different methods
- Modify the SQL to pass a Variable.

Transformation

This demonstration illustrates the different options for passing parameters.



Example 1

In this example, the Parameters are passed in a single data row.

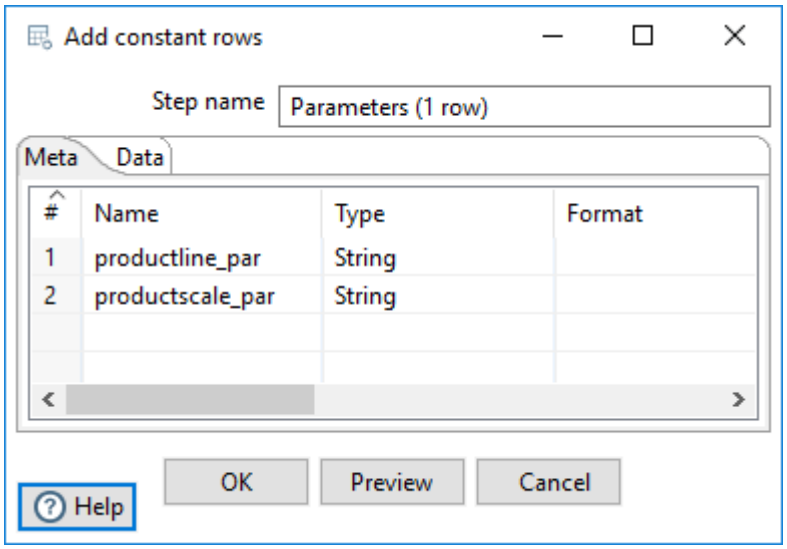
Data Grid

The Data Grid step allows you to enter a static list of rows in a grid. This is usually done for testing, reference or demo purposes.

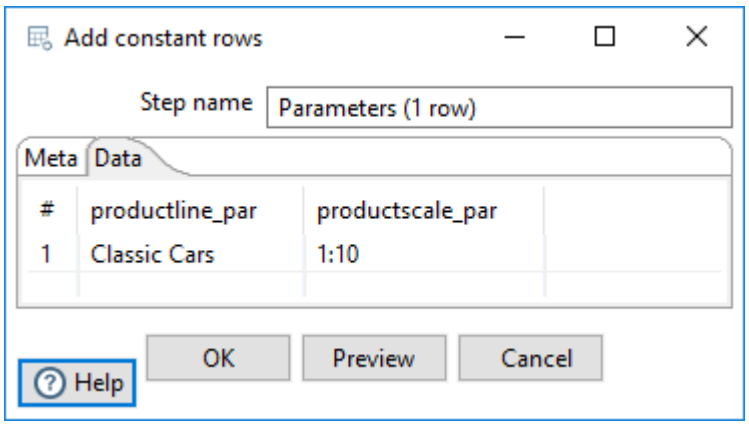
Options

- **Meta tab:** You can specify the field metadata (output specification) of the data
- **Data tab:** This grid contains the data. Everything is entered in String format so make sure you use the correct format masks in the metadata tab.

1. Drag the Data Grid step onto the canvas.
2. Open the Data Grid properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click on the Data tab, and configure the following:

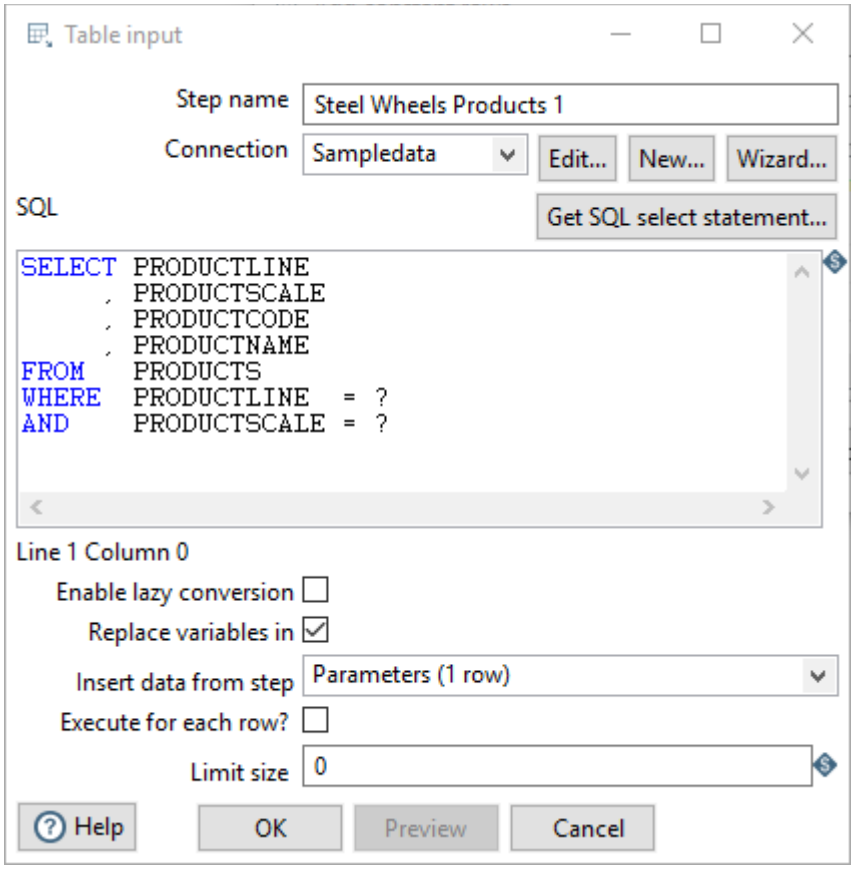


5. Click OK.

Table Input

This step is used to read information from a database, using a connection and SQL. Basic SQL statements can be generated automatically by clicking ***Get SQL select statement***.

1. Drag the Table Input step onto the canvas.
2. Open the Table Input properties dialog box.
3. Ensure the following details are configured, as outlined below:



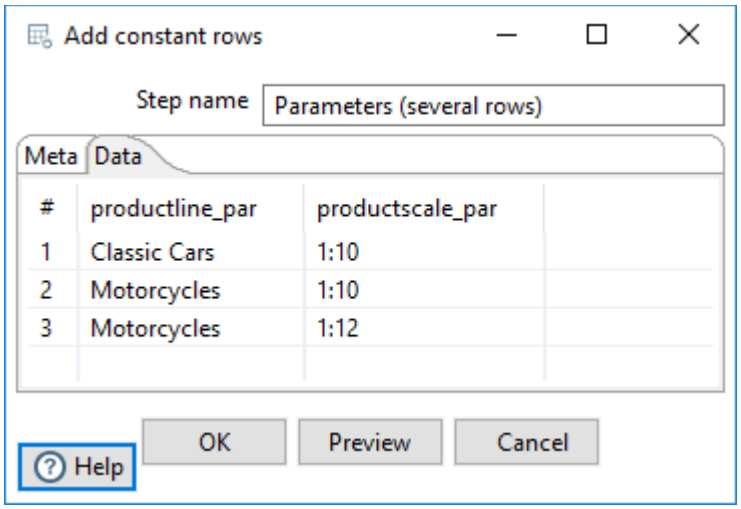
4. Click OK.

- The two database variables ? for PRODUCTLINE and PRODUCTSCALE are replaced with the values set in the Parameters (1 row) step.

Example 2

In this example, another data row is added.

1. Copy / Paste the previous Example, and rename the various Steps.
2. Open the Data Grid step and add the following data row:

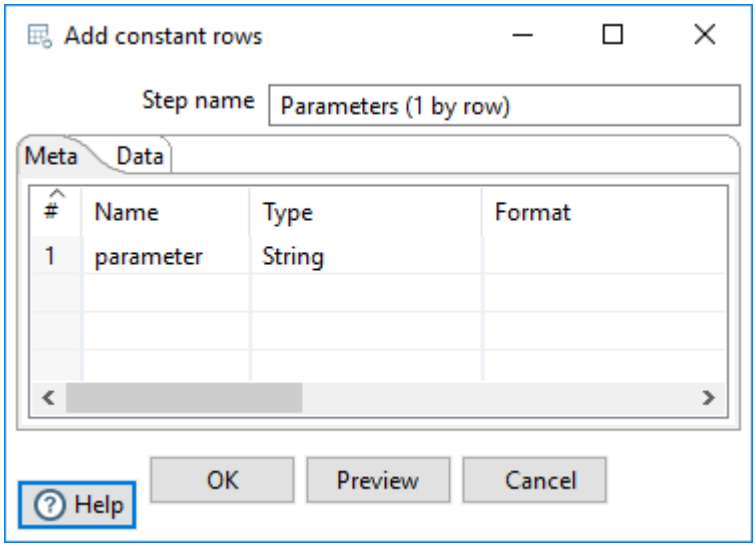


3. Click OK.

Example 3

A single parameter passes the required values.

1. Copy / Paste the previous Example, and rename the various Steps.
2. Open the Data Grid step and configure as illustrated below:

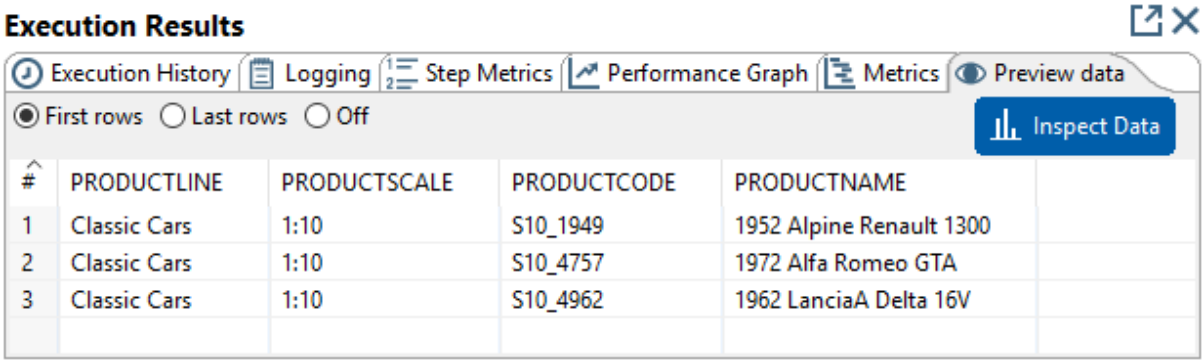


3. Click OK.

RUN the Transformation

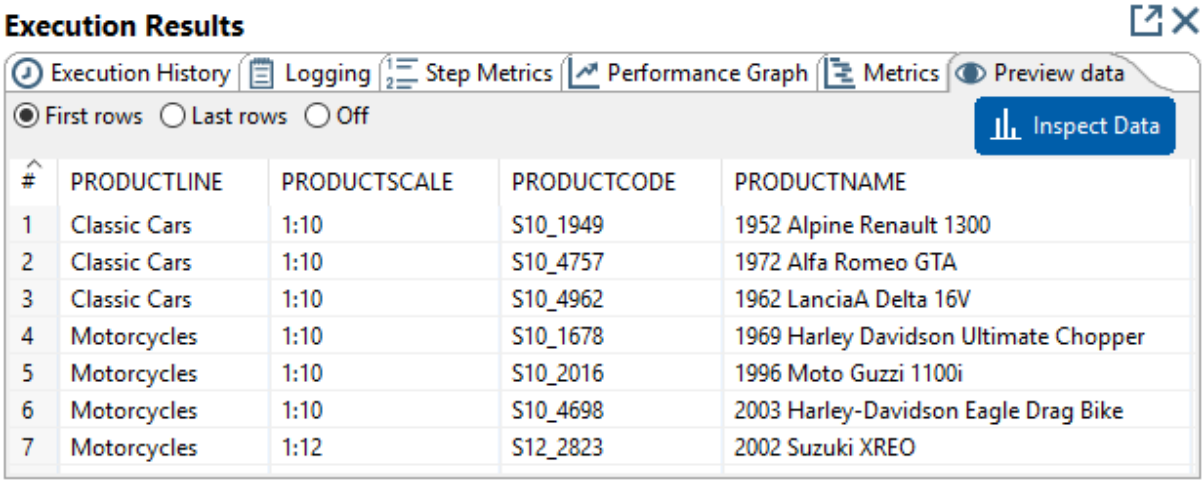
1. Click the Run button in the Canvas Toolbar.
2. Click on the Preview tab:

Example 1



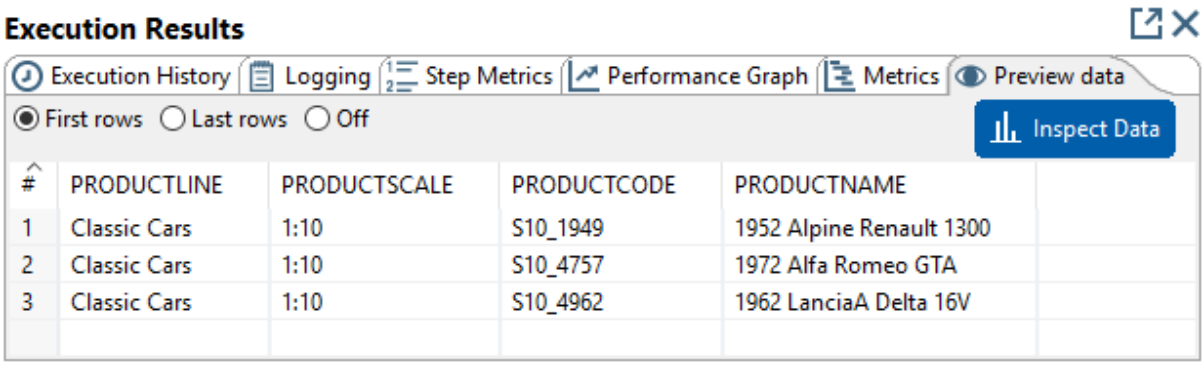
#	PRODUCTLINE	PRODUCTSCALE	PRODUCTCODE	PRODUCTNAME
1	Classic Cars	1:10	S10_1949	1952 Alpine Renault 1300
2	Classic Cars	1:10	S10_4757	1972 Alfa Romeo GTA
3	Classic Cars	1:10	S10_4962	1962 LanciaA Delta 16V

Example 2



#	PRODUCTLINE	PRODUCTSCALE	PRODUCTCODE	PRODUCTNAME
1	Classic Cars	1:10	S10_1949	1952 Alpine Renault 1300
2	Classic Cars	1:10	S10_4757	1972 Alfa Romeo GTA
3	Classic Cars	1:10	S10_4962	1962 LanciaA Delta 16V
4	Motorcycles	1:10	S10_1678	1969 Harley Davidson Ultimate Chopper
5	Motorcycles	1:10	S10_2016	1996 Moto Guzzi 1100i
6	Motorcycles	1:10	S10_4698	2003 Harley-Davidson Eagle Drag Bike
7	Motorcycles	1:12	S12_2823	2002 Suzuki XREO

Example 3

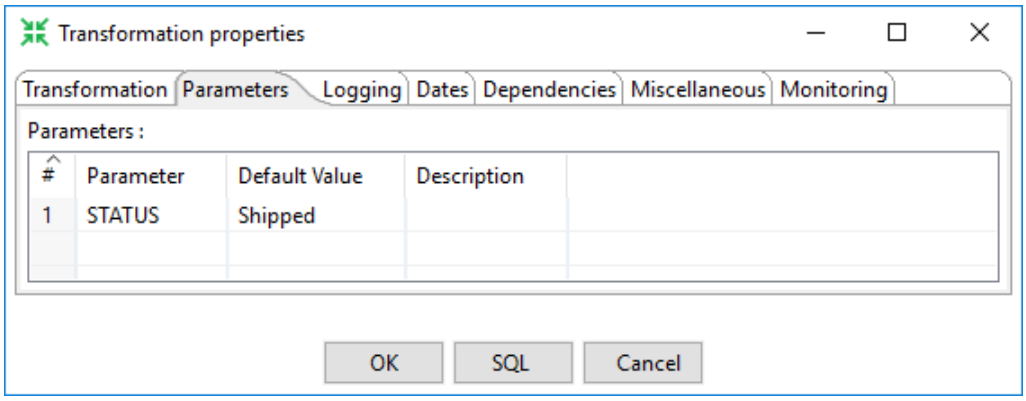


#	PRODUCTLINE	PRODUCTSCALE	PRODUCTCODE	PRODUCTNAME
1	Classic Cars	1:10	S10_1949	1952 Alpine Renault 1300
2	Classic Cars	1:10	S10_4757	1972 Alfa Romeo GTA
3	Classic Cars	1:10	S10_4962	1962 LanciaA Delta 16V

Variables

Part II of the Demo illustrates how to set up a Variable, illustrated with Guide Demo 3-2-3_parameters_orders_variables.ktr.

1. Open GD 3-2-3_parameters_orders_variables.ktr
2. Double-click on the canvas, to open Transformation Properties.
3. Click on the Parameters tab, and configure as illustrated below:

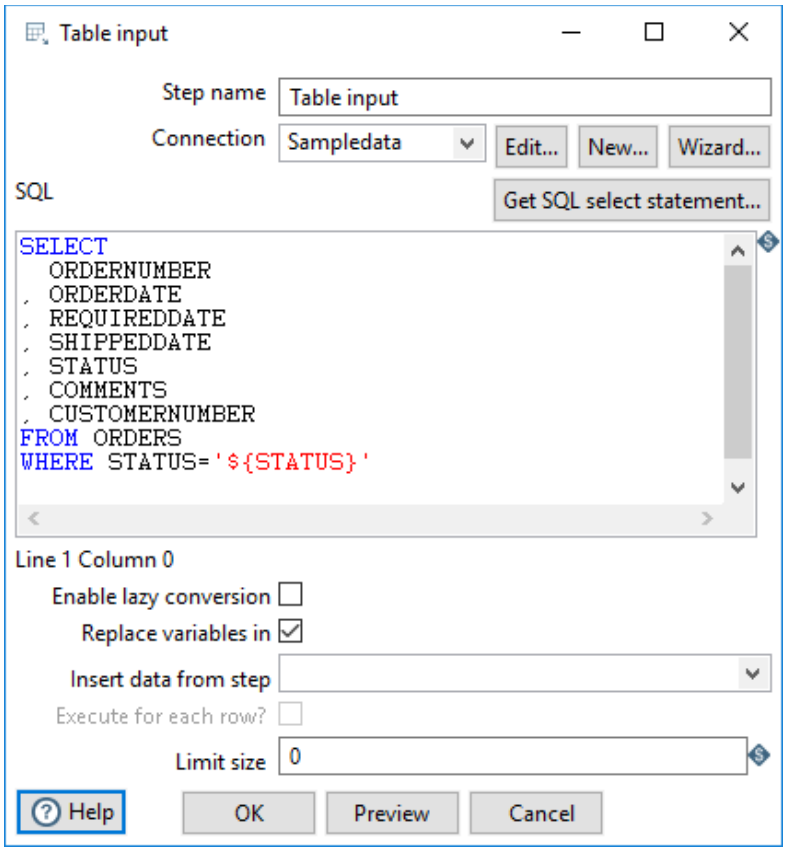


4. Click OK.

- Sets the value for the STATUS parameter.

Modify the Table Input

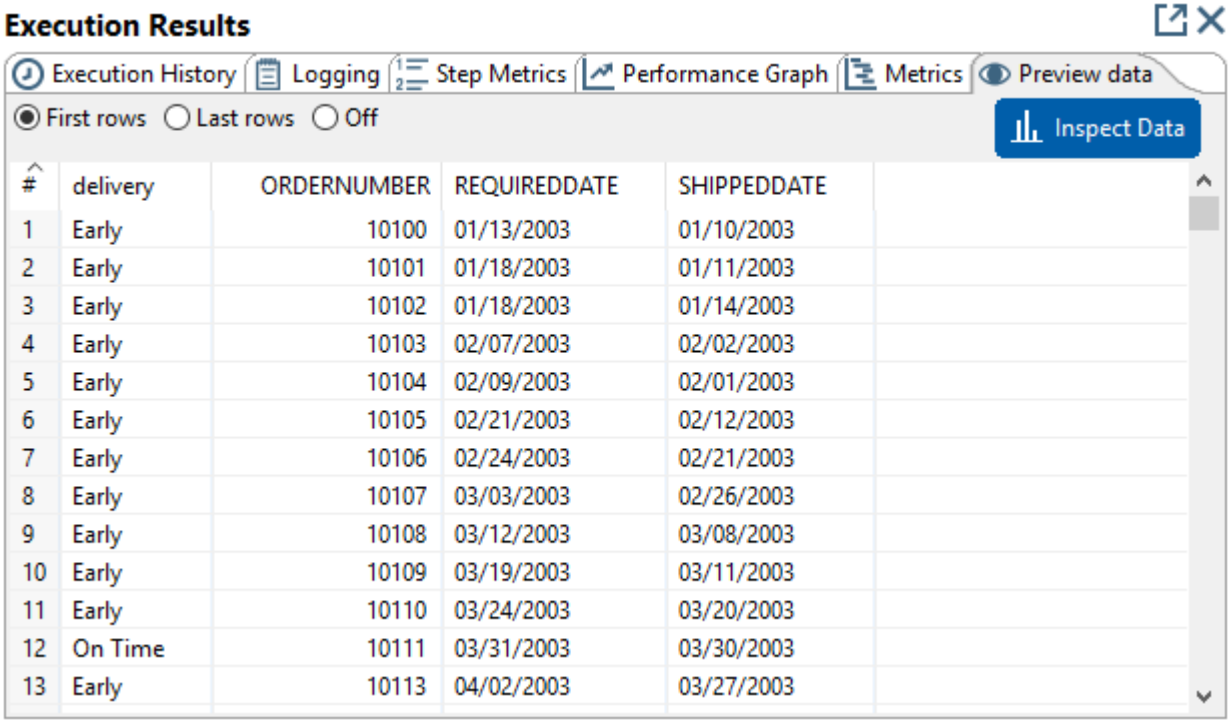
1. Double-click on the Table Input step.
2. Modify the SQL statement as illustrated below:



3. Click OK.

RUN the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Click on the Preview tab:



#	delivery	ORDERNUMBER	REQUIREDDATE	SHIPPEDDATE
1	Early	10100	01/13/2003	01/10/2003
2	Early	10101	01/18/2003	01/11/2003
3	Early	10102	01/18/2003	01/14/2003
4	Early	10103	02/07/2003	02/02/2003
5	Early	10104	02/09/2003	02/01/2003
6	Early	10105	02/21/2003	02/12/2003
7	Early	10106	02/24/2003	02/21/2003
8	Early	10107	03/03/2003	02/26/2003
9	Early	10108	03/12/2003	03/08/2003
10	Early	10109	03/19/2003	03/11/2003
11	Early	10110	03/24/2003	03/20/2003
12	On Time	10111	03/31/2003	03/30/2003
13	Early	10113	04/02/2003	03/27/2003

Summary

Topics covered in this section:

- Overview of Steel Wheels Database
- Connecting to databases
- Previewing and getting data from a database
- Inserting, updating, and deleting data from a database

Pentaho ships with a sample data for a fictional store named Steel Wheels. This data is stored in a database that is going to be the starting point for you to learn how to work with databases in PDI.

Once connected, you have a huge number of steps available that enable Ralph Kimball changes to the tables.

Type 1 - Overwriting the old value. In this method, no history of dimension changes is kept in the database. The old dimension value is simply overwritten with the new one. This type is easy to maintain and is often use for data which changes are caused by processing corrections (e.g. removal special characters, correcting spelling errors).

Type 2 - Creating a new additional record. In this methodology, all history of dimension changes is kept in the database. You capture attribute change by adding a new row with a new surrogate key to the dimension table. Both the prior and new rows contain as attributes the natural key (or another durable identifier). Also 'effective date' and 'current indicator' columns are used in this method. There could be only one record with current indicator set to 'Y'. For 'effective date' columns, i.e. start_date and end_date, the end_date for current record usually is set to value 9999-12-31. Introducing changes to the dimensional model in type 2 could be very expensive database operation so it is not recommended to use it in dimensions where a new attribute could be added in the future.

You also configured the following key steps:

Analytic Query - This step allows you to 'scroll' forward and backwards across rows.

Calculator - This calculator step provides you with predefined functions that can be executed on input field values. If need other generic, often used functions, visit the Pentaho community page and let Pentaho know about your enhancement request.

Combination Lookup / Update - The Combination Lookup-Update step allows you to store information in a junk-dimension table, and can possibly also be used to maintain Kimball pure Type 1 dimensions.

Database Value Lookup - The Database lookup step allows you to look up values in a database table. Lookup values are added as new fields onto the stream.

Delete - Operates in the same way as Truncate. Deletes rows.

Dimension Lookup / Update - The Dimension Lookup/Update step allows you to implement Ralph Kimball's slowly changing dimension for both types: Type I (update) and Type II (insert) together with some additional functions.

Get variables - This step allows you to get the value of a variable. This step can return rows or add values to input rows.

Group By - This step allows you to calculate values over a defined group of fields. Examples of common use cases are: calculate the average sales per product or get the number of yellow shirts that we have in stock.

Insert / Update - The Insert/Update step first looks up a row in a table using one or more lookup keys. If the row can't be found, it inserts the row. If it can be found and the fields to update are the same, nothing is done. If they are not all the same, the row in the table is updated.

Number ranges - Create ranges based on numeric fields.

Sort Rows - The Sort rows step sorts rows based on the fields you specify and on whether they should be sorted in ascending or descending order.

Switch Case - What this step does is implement the Switch/Case statement found in popular programming languages like Java.

Table Output - The Table Output step allows you to load data into a database table. Table Output is equivalent to the DML operator INSERT. This step provides configuration options for target table and a lot of housekeeping and/or performance-related options such as Commit Size and Use batch update for inserts.

Table Input - This step is used to read information from a database, using a connection and SQL. Basic SQL statements can be generated automatically by clicking *Get SQL select statement*.

Update - The Update step first looks up a row in a table using one or more lookup keys. If the row matches the lookups, then it updates the record.

4: Data Enrichment

Topics covered in this section:

- Stream Operations
  - Merge rows
  - Merge rows (diff)
- Lookups
- Joins
  - Join rows
  - Merge Join
  - Database Join
- Scripting
  - Formula
  - Modified JavaScript Value

Data Enrichment is a value adding process, where external data from multiple sources is added to the existing data set to enhance the quality and richness of the data. This process provides more information of the product / service to the customer.

A common data enrichment process could, for example, correct likely misspellings or typographical errors in a database using precision algorithms. Following this logic, data enrichment could also add information to simple data tables.

Another way that data enrichment can work is in extrapolating data. Through methodologies such as fuzzy logic, engineers can produce more from a given raw data set. This and other projects can be described as data enrichment activities.

There are numerous data enhancement options available including:

- Telephone & Fax numbers
- Additional contact names
- Residential or Business location
- Standard Industrial Classification Codes (SIC's) & 'Market Sector' codes
- No. of employees
- Small office /Home office (SoHo's)
- Household income /Age
- Credit score
- Financial information
- Adding valuable geographic information and mapping (GIS) such as location analysis, distance calculations, spatial analysis, natural boundary analysis, and more
- Enhancing data by classifying, segmenting, and aggregating customer data using advanced statistical methodologies such as factor, cluster, and conjoint analysis

Introduction

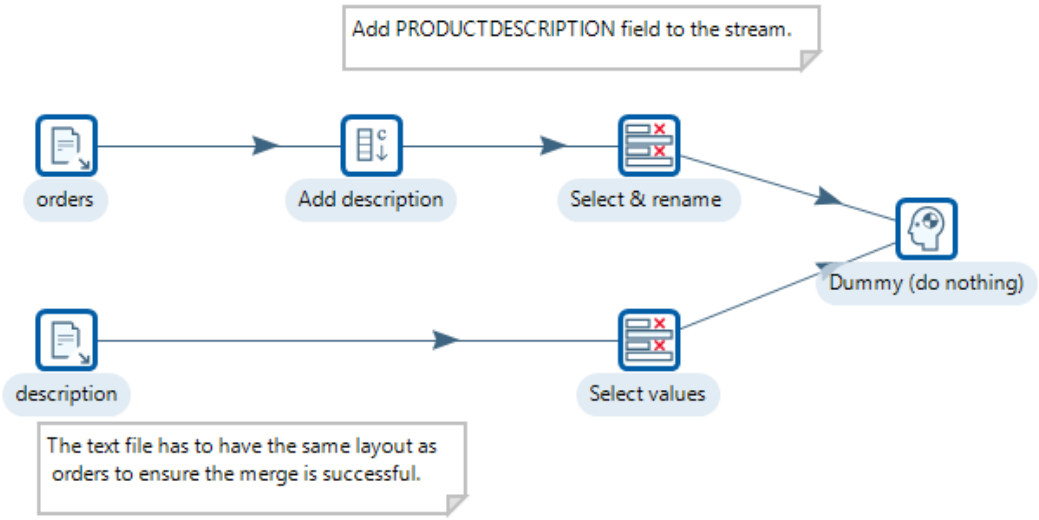
The Transformation underlines the 'rules' for manipulating data streams. Each data stream must have the same structure / layout, before they can be merged.

Objectives

In this guided demonstration, you will:

- Configure the following steps:
  - Add constant

Transformation



By now you should have a good idea on how to configure some of the steps we've previously covered.

Text File Inputs

The Text File Input step is used to read data from a variety of different text-file types. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

The Text File Input step provides you with the ability to specify a list of files to read, or a list of directories with wild cards in the form of regular expressions. In addition, you can accept filenames from a previous step making filename handling more even more generic.

1. Examine both the orders.txt and description.txt.
2. Configure the Text file input steps to point to, and retrieve the data from each of the files.

Add Constant

The Add constant values step is a simple and high performance way to add constant values to the stream.

1. The PRODUCTDESCRIPTION field is added to the 'orders stream' to ensure the data stream matches the 'description' data stream.
- This ensures both data streams have the same structure / layout.

Select values

The Select Values step is useful for selecting, removing, renaming, changing data types and configuring the length and precision of the fields on the stream. These operations are organized into different categories:

- Select and Alter — Specify the exact order and name in which the fields could be placed in the output rows
- Remove — Specify the fields that could be removed from the output rows
- Meta-data - Change the name, type, length and precision (the metadata) of one or more fields

Each of the Select values ensures that each data stream is consistent in layout before merging. Each field must be in the correct order within the data stream so that mappings are successful.

RUN the Transformation

1. Run the Transformation.
2. Click on the Dummy step and 'Preview'.

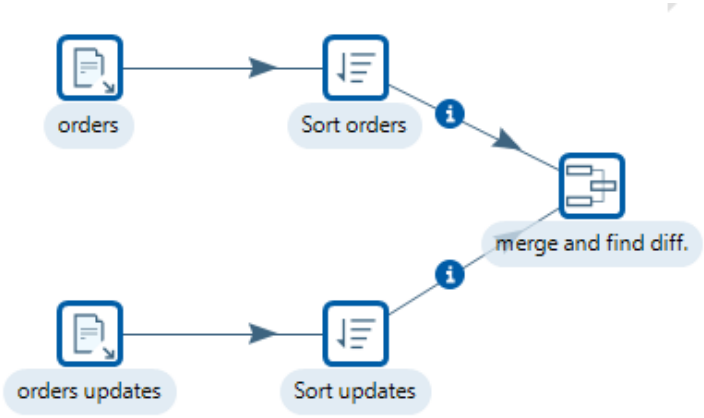
Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

First rows | Last rows | Off — Inspect Data

#	PRODUCTCODE	PRODUCTNAME	PRODUCTLINE	PRODUCTSCALE	PRODUCTVENDOR
1	S10_1678	1969 Harley Davidson Chopper	Motorcycles	1:10	Min Lin Diecast
2	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations
3	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics
4	S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast
5	S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics
6	S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast
7	S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design
8	S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast
9	S12_1666	1958 Setra Bus	Trucks and Buses	1:12	Welly Diecast Productions
10	S10_1678	<null>	<null>	<null>	<null>
11	S10_1949	<null>	<null>	<null>	<null>
12	S10_2016	<null>	<null>	<null>	<null>
13	S10_4698	<null>	<null>	<null>	<null>
14	S10_4757	<null>	<null>	<null>	<null>
15	S10_4962	<null>	<null>	<null>	<null>
16	S12_1099	<null>	<null>	<null>	<null>
17	S12_1108	<null>	<null>	<null>	<null>
18	S12_1666	<null>	<null>	<null>	<null>

- The resulting data streams are merged; however, redundancy has occurred.

Introduction	The Merge row (diff) compares the values between the merging rows and sets a 'flag'.
Objectives	In this guided demonstration, you will: • Configure the following steps: ○ Merge rows (diff)
Transformation	

Text File Input

1. Compare the difference between the orders.txt and the orders_update.txt

   New records

   S10_1949;1952 Alpine Renault 1300;Classic Cars;1:10;Classic Metal Creations;Turnable front wheels, steering function, detailed interior, detailed engine, opening hood, opening trunk, opening doors, and detailed chassis.

   S12_1108;2001 Ferrari Enzo;Classic Cars;1:12;Second Gear Diecast;Turnable front wheels, steering function, detailed interior, detailed engine, opening hood, opening trunk, opening doors, and detailed chassis.

   Deleted records

   S10_4757;1972 Alfa Romeo GTA;Classic Cars;1:10;Motor City Art Classics;Features include: Turnable front wheels, steering function, detailed interior, detailed engine, opening hood, opening trunk, opening doors, and detailed chassis.

   Changed records

   S10_4698;2004 Harley-Davidson Eagle Drag Bike;Motorcycles;1:10;Red Start Diecast;Model features, official Harley Davidson logos and insignias, detachable rear wheelie bar, heavy diecast metal with resin parts, authentic multi-color tampo-printed graphics, separate engine drive belts, free-turning front
   fork.

Merge rows (diff)

Merge rows allows you to compare two streams of rows. This is useful for comparing data from two different times. It is often used in situations where the source system of a data warehouse does not contain a date of last update.

The two sets of records, an original stream (the existing data) and the new stream (the new data), are compared and merged.

Only the last version of a record is passed to the next steps each time. The row is marked as follows:

"identical" - The key was found in both streams and the values in both fields are identical;

"changed" - The key was found in both streams but one or more values are different;

"new" - The key was not found in the original stream;

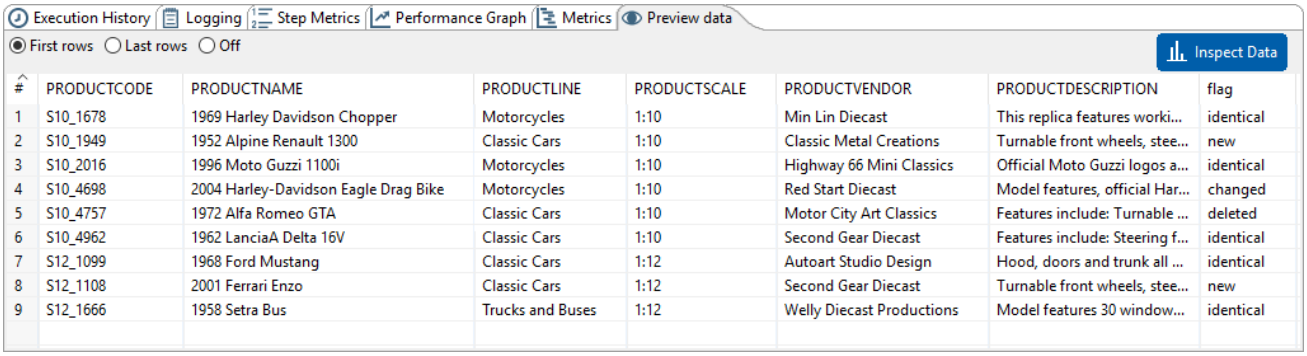
"deleted" - The key was not found in the new stream.

The records coming from the new stream are passed on to the next steps, except when it is "deleted".

Both streams must be sorted on the specified key(s).

RUN the Transformation

1. Run the Transformation.
2. Examine and compare the records.



#	PRODUCTCODE	PRODUCTNAME	PRODUCTLINE	PRODUCTSCALE	PRODUCTVENDOR	PRODUCTDESCRIPTION	flag
1	S10_1678	1969 Harley Davidson Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features worki...	identical
2	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels, stee...	new
3	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi logos a...	identical
4	S10_4698	2004 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast	Model features, official Har...	changed
5	S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics	Features include: Turnable ...	deleted
6	S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast	Features include: Steering f...	identical
7	S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design	Hood, doors and trunk all ...	identical
8	S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast	Turnable front wheels, stee...	new
9	S12_1666	1958 Setra Bus	Trucks and Buses	1:12	Welly Diecast Productions	Model features 30 window...	identical

Introduction

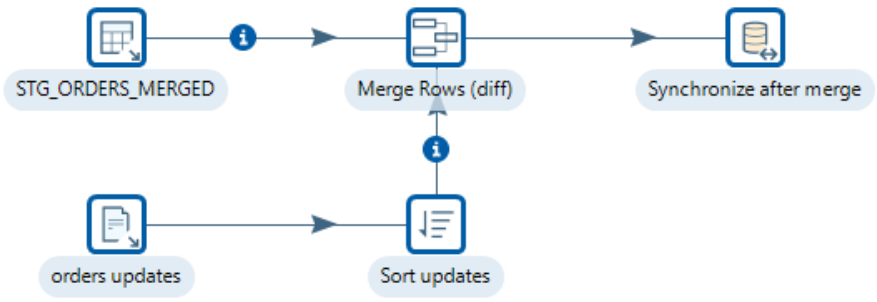
If the Merge rows (diff) step is used in conjunction with the Synchronize after merge, then based on the value of the flag, the database can be updated.

Objectives

In this guided demonstration, you will:

- Configure the following steps:
  - Synchronize after merge

Transformation



Synchronize after merge

This step can be used in conjunction with the Merge Rows (diff) transformation step. The Merge Rows (diff) transformation step appends a Flag column to each row, with a value of "identical", "changed", "new" or "deleted". This flag column is then used by the **Synchronize after merge** transformation step to carry out updates/inserts/deletes on a connection table.

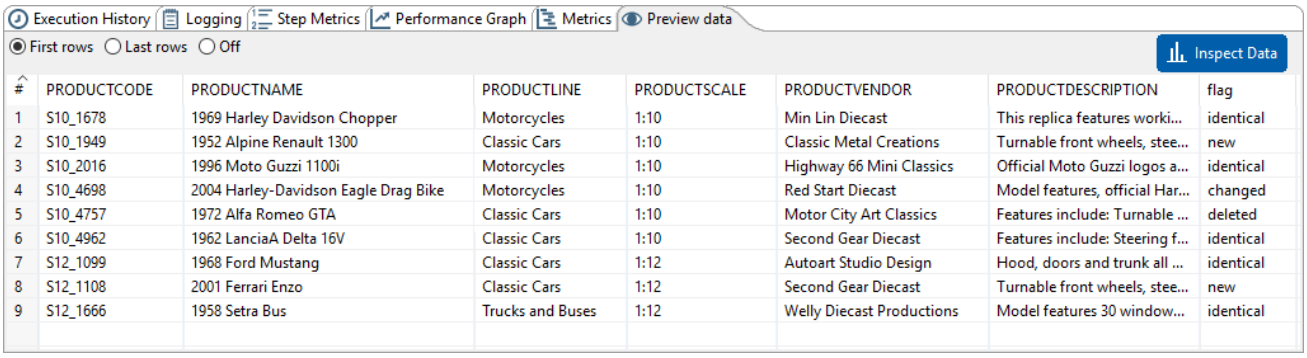
This step uses the flag value to perform the sync operations on the database table.

Advanced Tab

Option	Description	Default value
Operation fieldname	**This is a required field.** This field is used by the step to obtain an operation flag for the current row.	"flagfield"
Insert when value equal	Specify the value of the Operation fieldname which signifies that an Insert should be carried out.	"new"
Update when value equal	Specify the value of the Operation fieldname which signifies that an Update should be carried out.	"changed"
Delete when value equal	Specify the value of the Operation fieldname which signifies that a Delete should be carried out.	"deleted"
Perform lookup	Performs a lookup when deleting or updating. If the lookup field is not found, then an exception is thrown. This option can be used as an extra check if you wish to check updates/deletes prior to their execution.	Not applicable

RUN the Transformation

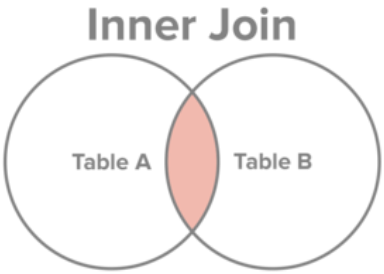
1. Run the Transformation.
2. Examine and compare the records.



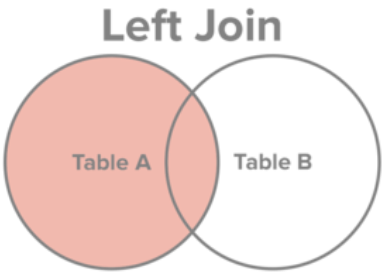
#	PRODUCTCODE	PRODUCTNAME	PRODUCTLINE	PRODUCTSCALE	PRODUCTVENDOR	PRODUCTDESCRIPTION	flag
1	S10_1678	1969 Harley Davidson Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features worki...	identical
2	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels, stee...	new
3	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi logos a...	identical
4	S10_4698	2004 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast	Model features, official Har...	changed
5	S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics	Features include: Turnable ...	deleted
6	S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast	Features include: Steering f...	identical
7	S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design	Hood, doors and trunk all ...	identical
8	S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast	Turnable front wheels, stee...	new
9	S12_1666	1958 Setra Bus	Trucks and Buses	1:12	Welly Diecast Productions	Model features 30 window...	identical

There are four basic types of SQL joins: inner, left, right, and full. The easiest and most intuitive way to explain the difference between these four types is by using a Venn diagram, which shows all possible logical relations between data sets.

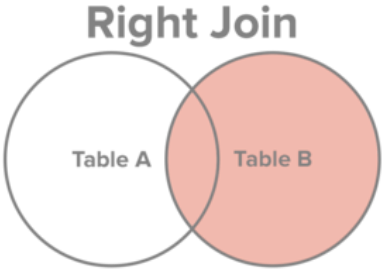
Again, it's important to stress that before you can begin using any join type, you'll need to extract the data and load it into an RDBMS, where you can query tables from multiple sources.



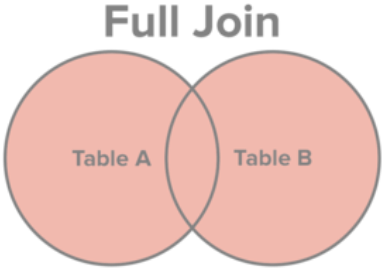
Select all records from Table A and Table B, where the join condition is met.



Select all records from Table A, along with records from Table B for which the join condition is met (if at all).



Select all records from Table B, along with records from Table A for which the join condition is met (if at all).



Select all records from Table A and Table B, regardless of whether the join condition is met or not.

Guided Demo 4-2-1: Cross Join

Introduction

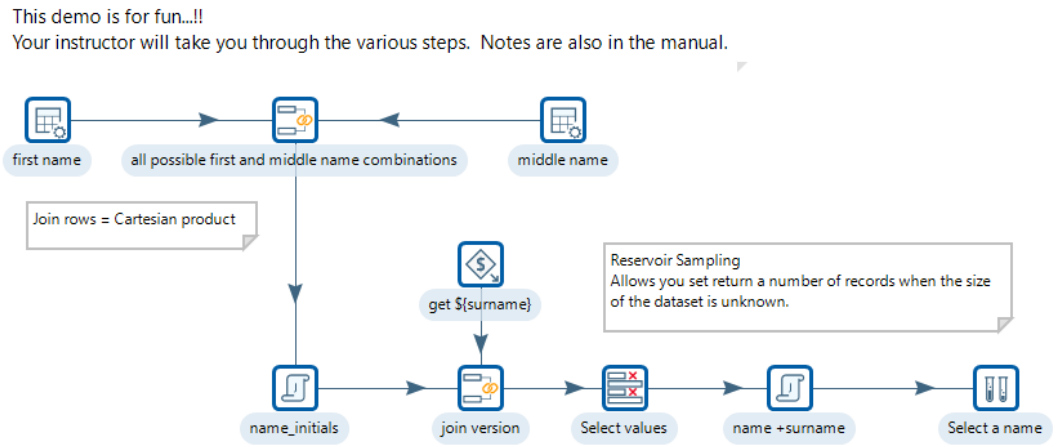
A Cross Join, is a **join** of every row of one table to every row of another table. This normally happens when no matching **join** columns are specified.

Objectives

In this guided demonstration, you will:

- Configure the following steps:
  - Join
  - Reservoir Sampling

Transformation



This demonstration highlights the use of the Join step.

Data Grid

You should be familiar with the Data Grid step. Used to list values for boys first and middle names.

Join

Joins all possible first_name, middle_name combinations together.

User Defined Java Expression

2 new fields are added to the data stream:

- first_middle_name: concates the first_names and middle names.
- initials: returns the 2 character initials.

  This method has two variants and returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string or up to endIndex – 1, if the second argument is given.

Get Variable

Returns the value associated with the ${surname}

Join

Joins all possible first_middle_name, surname combinations together. The output for initials is also excludes in the list various initials combinations.

Select Values

Rearranges / determines the stream data fields.

User Defined Java Expression

2 new fields are added to the data stream:

- baby_initials: returns the babys' 3 character initials.
- baby_name: concates first_name + middle_name + surname

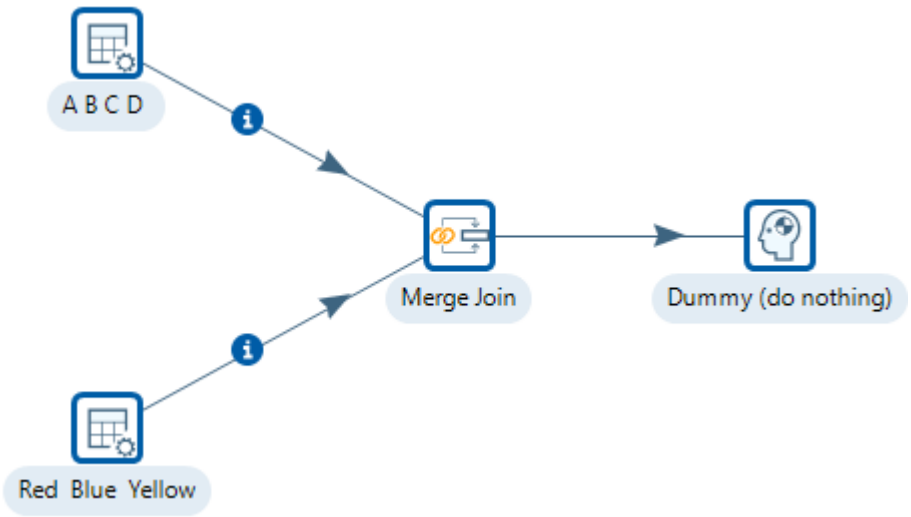
Reservoir Sampling

Returns 5 sampled records.

Returns Random seed ${seed}

Reservoir Sampling allows you to select a set number of random records, from an unknown number 'reservoir' of records, i.e. not known beforehand.

Use a different seed value to ensure no two 'sets' are the same.

Introduction	A guided demo to illustrate the various joins.
Objectives	In this guided demonstration, you will: • Configure the following steps: ○ Merge Join
Transformation	

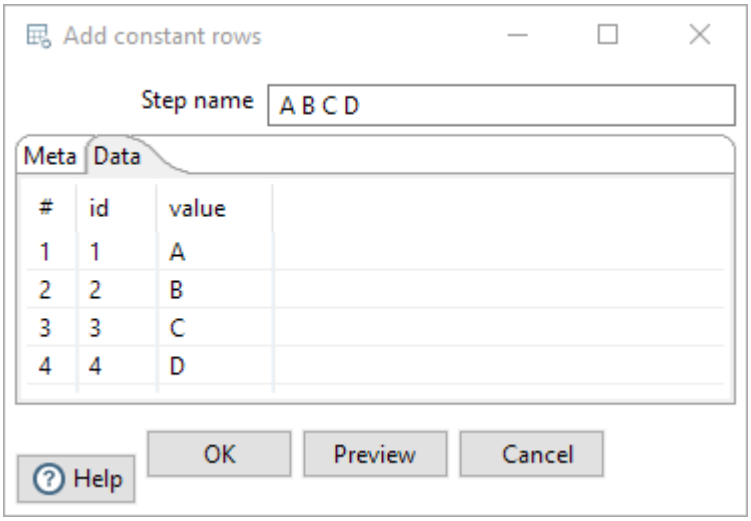
Data Grid - A B C D

The Data Grid step allows you to enter a static list of rows in a grid. This is usually done for testing, reference or demo purposes.

Options

- **Meta tab:** You can specify the field metadata (output specification) of the data
- **Data tab:** This grid contains the data. Everything is entered in String format so make sure you use the correct format masks in the metadata tab.

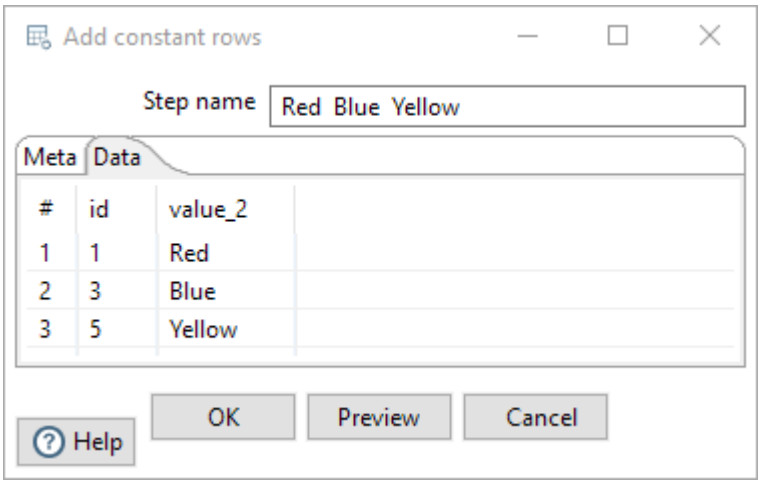
1. Drag the Data Grid step onto the canvas.
2. Open the Data Grid properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click OK.

Data Grid – Red Blue Yellow

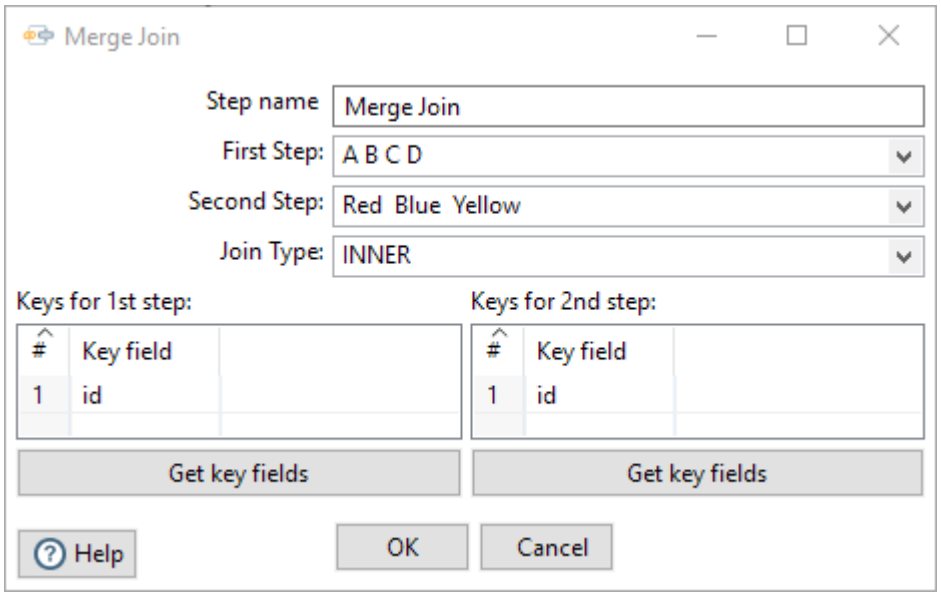
1. Drag the Data Grid step onto the canvas.
2. Open the Data Grid properties dialog box.
3. Ensure the following details are configured, as outlined below:



Merge Join

The Merge Join step performs a classic merge join between data sets with data coming from two different input steps. Join options include INNER, LEFT OUTER, RIGHT OUTER, and FULL OUTER.

1. Drag the Merge Join step onto the canvas.
2. Open the Merge Join properties dialog box.
3. Select various Join Types to view the resulting dataset



Dummy

1. Highlight the step to view the various 'Join Type' datasets.

Introduction

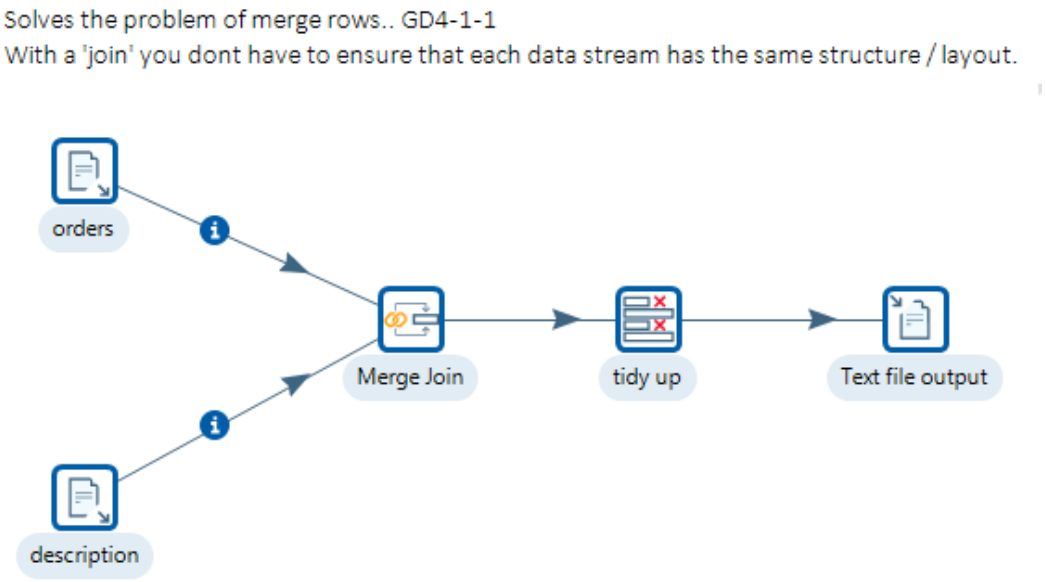
Back to our favourite Demonstration. Using Merge Join is a much more flexible solution to the constraints faced in GD4-1-1.

Objectives

In this guided demonstration, you will:

- Configure the following steps:
  - Merge Join

Transformation



Text File Inputs

The Text File Input step is used to read data from a variety of different text-file types. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

The Text File Input step provides you with the ability to specify a list of files to read, or a list of directories with wild cards in the form of regular expressions. In addition, you can accept filenames from a previous step making filename handling more even more generic.

1. Examine both the orders.txt and description.txt.
2. Configure the Text file input steps to point to, and retrieve the data from each of the files.

Merge Join

The Merge Join step performs a classic merge join between data sets with data coming from two different input steps. Join options include INNER, LEFT OUTER, RIGHT OUTER, and FULL OUTER.

> In this step rows are expected in to be sorted on the specified key fields. When using the Sort step, this works fine. When you sorted the data outside of PDI, you may run into issues with the internal case sensitive/insensitive flag.
>
> **Note**: If the key fields have the same name (e.g. ID), a second key field called $key_1 (e.g. ID_1) will be created in the result.

Select values

The Select Values step is useful for selecting, removing, renaming, changing data types and configuring the length and precision of the fields on the stream. These operations are organized into different categories:

- Select and Alter — Specify the exact order and name in which the fields could be placed in the output rows
- Remove — Specify the fields that could be removed from the output rows
- Meta-data - Change the name, type, length and precision (the metadata) of one or more fields

Performs a little house keeping…!

Text File Output

The Text file output step is used to export data to text file format. This is commonly used to generate Comma Separated Values (CSV files) that can be read by spreadsheet applications. It is also possible to generate fixed width files by setting lengths on the fields in the fields tab.

Introduction

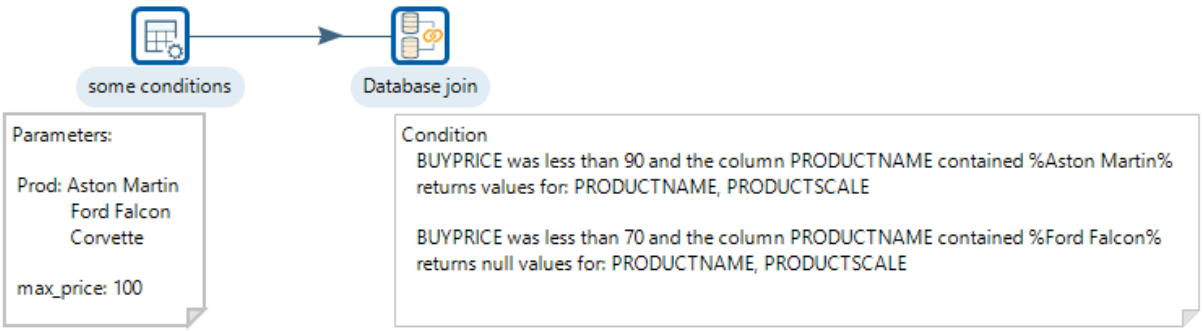
Searching for information in databases, text files, web services, and so on, is a very common task.

Objectives

In this guided demonstration, you will:

- Configure the following steps:
  - Database Join

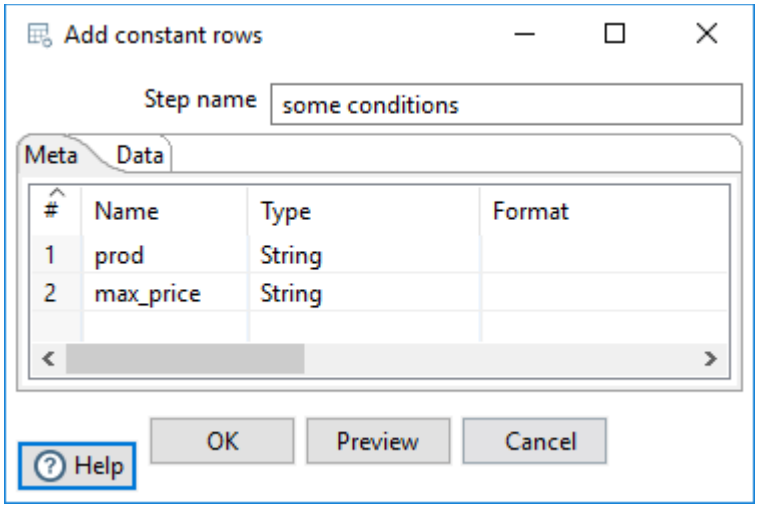
Transformation



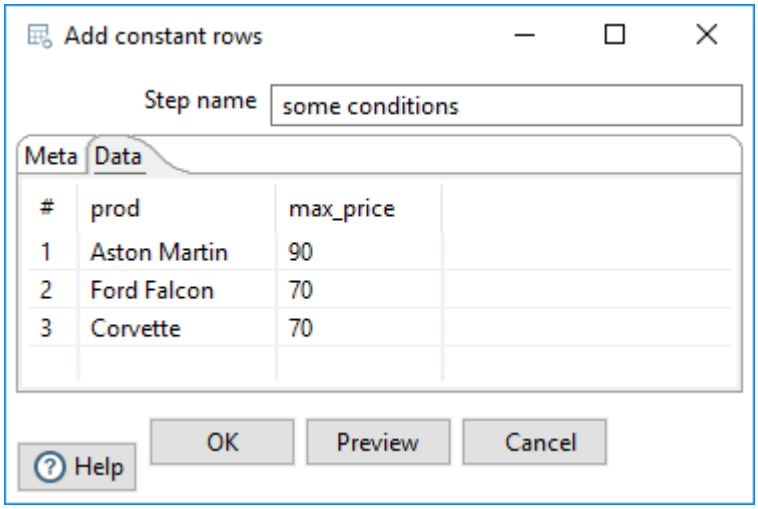
Data Grid

The Data Grid step allows you to enter a static list of rows in a grid. This is usually done for testing, reference or demo purposes.

1. Drag the Data Grid step onto the canvas.
2. Open the Data grid properties dialog box.
3. Ensure the following details are configured, as outlined below:



4. Click on the Data tab

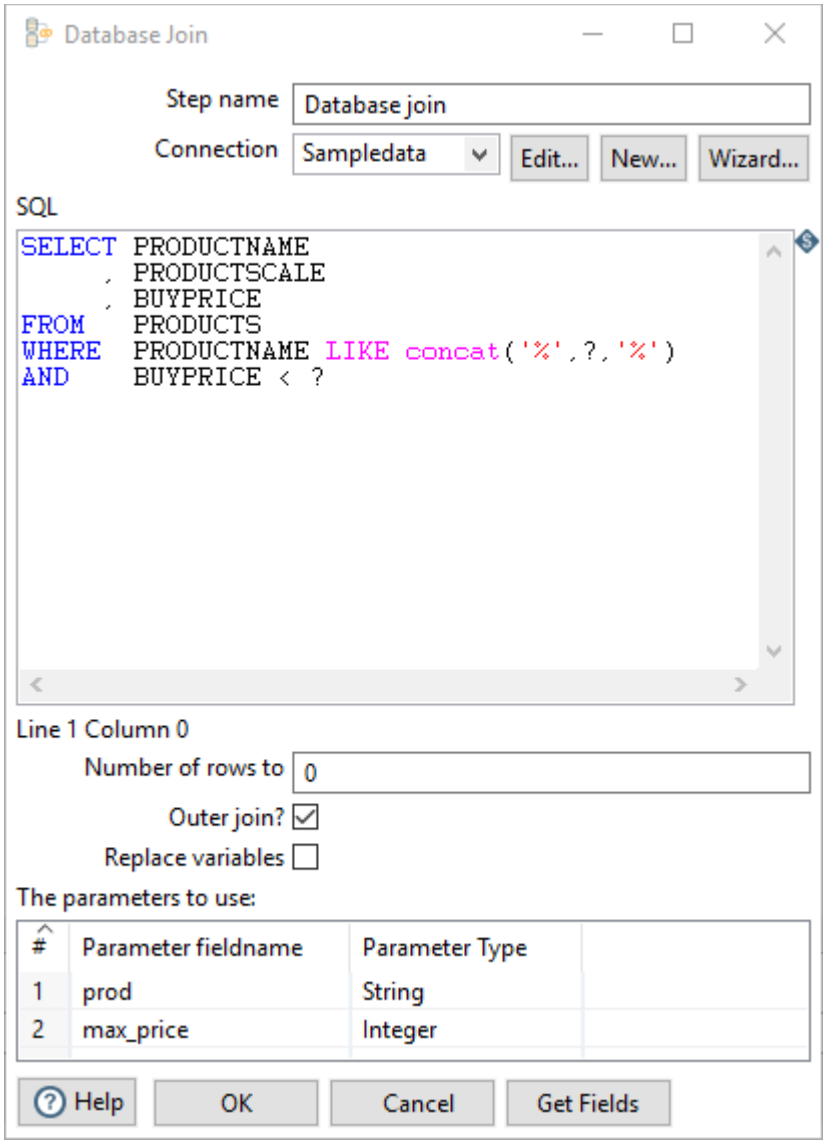


Database Join

The Database Join step allows you to run a query against a database using data obtained from previous steps. The parameters for this query are specified as follows:

- The data grid in the step properties dialog. This allows you to select the data coming in from the source hop.
- As question marks (?) in the SQL query. When the step runs, these will be replaced with data coming in from the fields defined from the data grid. The question marks will be replaced in the same order as defined in the data grid.

1. Drag the Database Join step onto the canvas.
2. Open the Database Join properties dialog box.
3. Ensure the following details are configured, as outlined below:



The 'Parameter fieldname' is where you specify the parameters, therefore the values, for the conditions. Each row in the grid represents a comparison between a column in the table, and a field in your stream, by using one of the provided comparators.

LIKE matches values.

You can't alias a column in the select clause and then use it in the where clause

The question marks you type in the SQL statement represent parameters. The purpose of these parameters is to be replaced with the fields you provide in 'Parameter fieldname'. For each row in the stream, the Database join step replaces the parameters in the same order as they are in the grid, and executes the SQL statement.

So, let's look at the WHERE conditions entered:

PRODUCTNAME LIKE like_statement and BUYPRICE < max_price

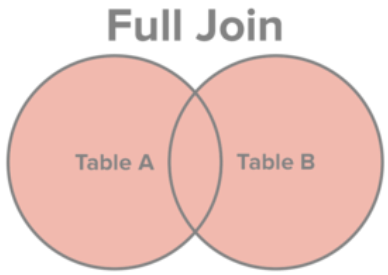
For the first record this translates as:

WHERE PRODUCTNAME LIKE concat ('%','Aston Martin','%') AND BUYPRICE < 90

As the Outer Join option is checked

The FULL OUTER JOIN keyword returns all rows from the left table and from the right table.

The FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.



The table dataset A is then compared with the kettle stream dataset B. If there's a match, then values for PRODUCTNAME and PRODUCTSCALE are returned.

This is not a database join. Instead of joining tables in a database, you are joining the result of a database query with a Kettle dataset.

For the second record:

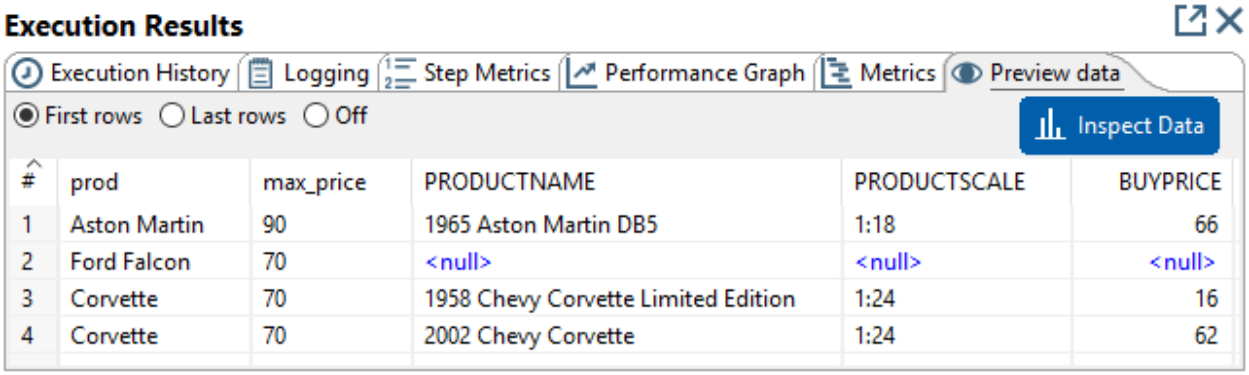
WHERE PRODUCTNAME LIKE concat ('%','Ford Falcon','%') AND BUYPRICE < 70

As there is no record, NULL values are returned for: PRODUCTNAME and PRODUCTSCALE.

So far, the results could be achieved using a Database Lookup step. However, there is a significant difference, as illustrated with the third row. For Corvette, the Database join found two matching rows in the database, and retrieved them both. Not possible with a Database lookup step.

Run and Save

1. Click the Run button in the Canvas Toolbar.
2. Click on the Preview tab:

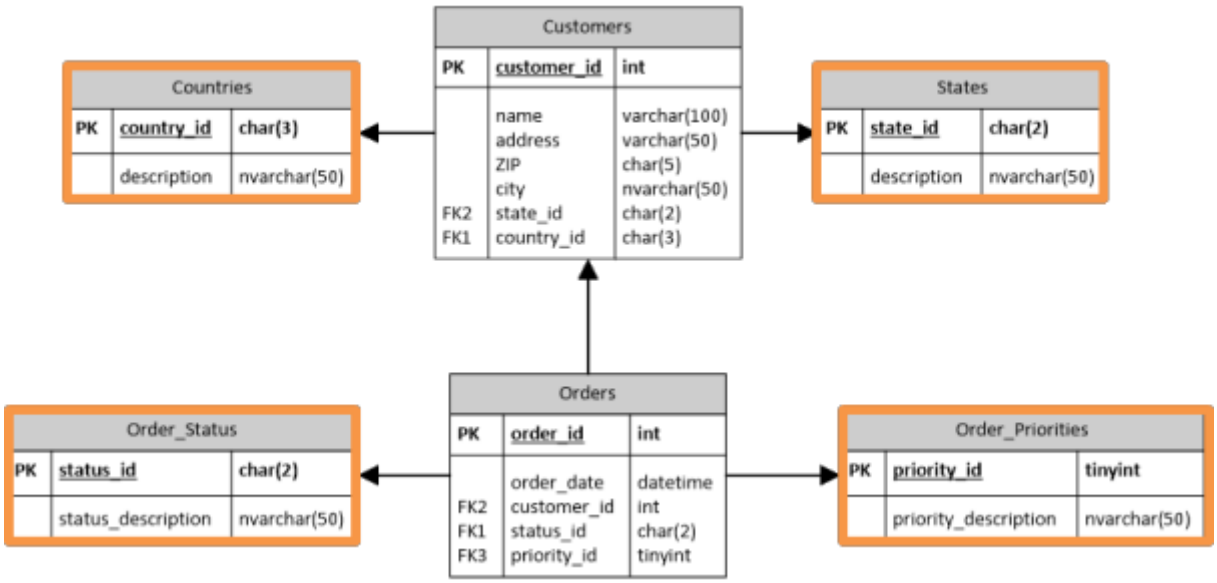


#	prod	max_price	PRODUCTNAME	PRODUCTSCALE	BUYPRICE
1	Aston Martin	90	1965 Aston Martin DB5	1:18	66
2	Ford Falcon	70	<null>	<null>	<null>
3	Corvette	70	1958 Chevy Corvette Limited Edition	1:24	16
4	Corvette	70	2002 Chevy Corvette	1:24	62

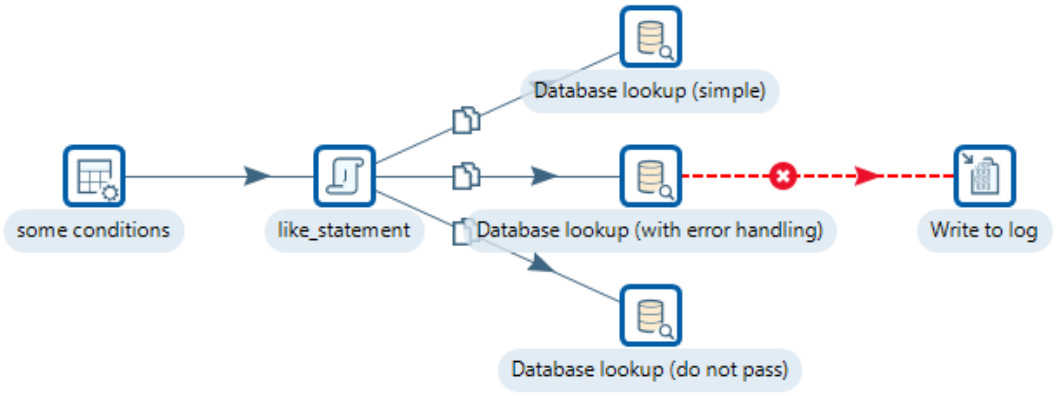
Besides transforming the data, you may need to search and bring data from other sources. Let us look at the following examples:

- You have some product codes and you want to look for their descriptions in an Excel file
- You have a value and want to get all products whose price is below that value from a database

Searching for information in databases, text files, web services, and so on, is a very common task, and Kettle has several steps for doing it.



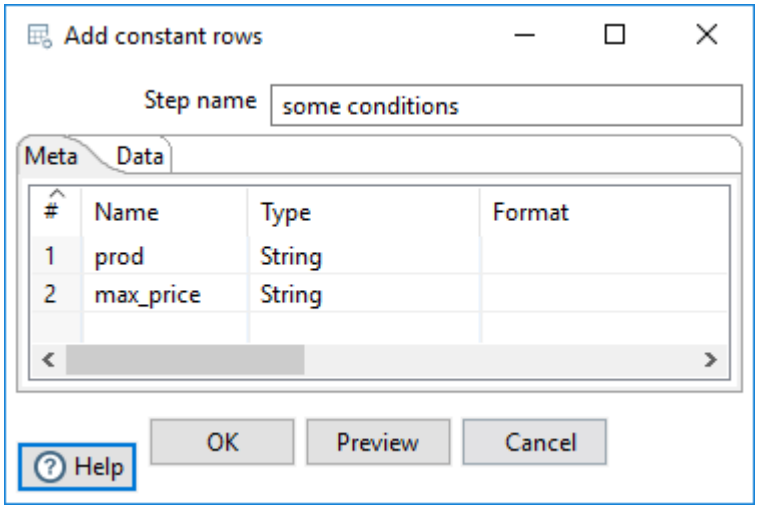
For instance, if your database is about sales, you probably have a Customers table and an Orders table, each with its own attributes resolved through a Foreign Key. The lookup tables are usually very small, with just a handful of rows in them.

Introduction	The Database lookup step allows you to look for values in a database table.
Objectives	In this guided demonstration, you will: • Configure the following steps: ○ Database Lookup
Transformation	

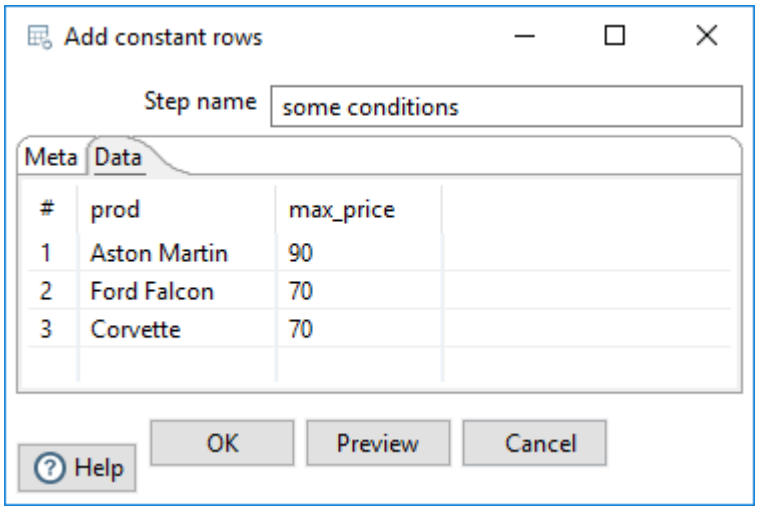
Data Grid

The Data Grid step allows you to enter a static list of rows in a grid. This is usually done for testing, reference or demo purposes.

1. Drag the Data Grid step onto the canvas.
2. Open the Data grid properties dialog box.
3. Ensure the following details are configured, as outlined below:



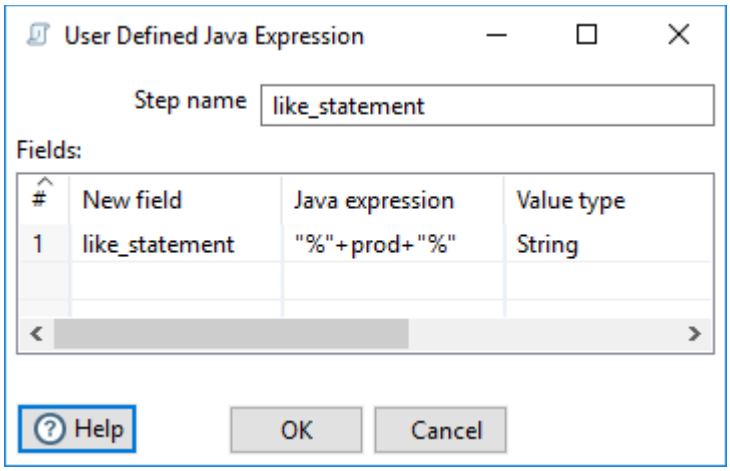
1. Click on the Data tab



User defined Java Expression

This step allows you to enter User Defined Java Expressions as a basis for the calculation of new values.

1. Drag the User Defined Java expression step onto the canvas.
2. Open the User defined Java expression properties dialog box.
3. Ensure the following details are configured, as outlined below:

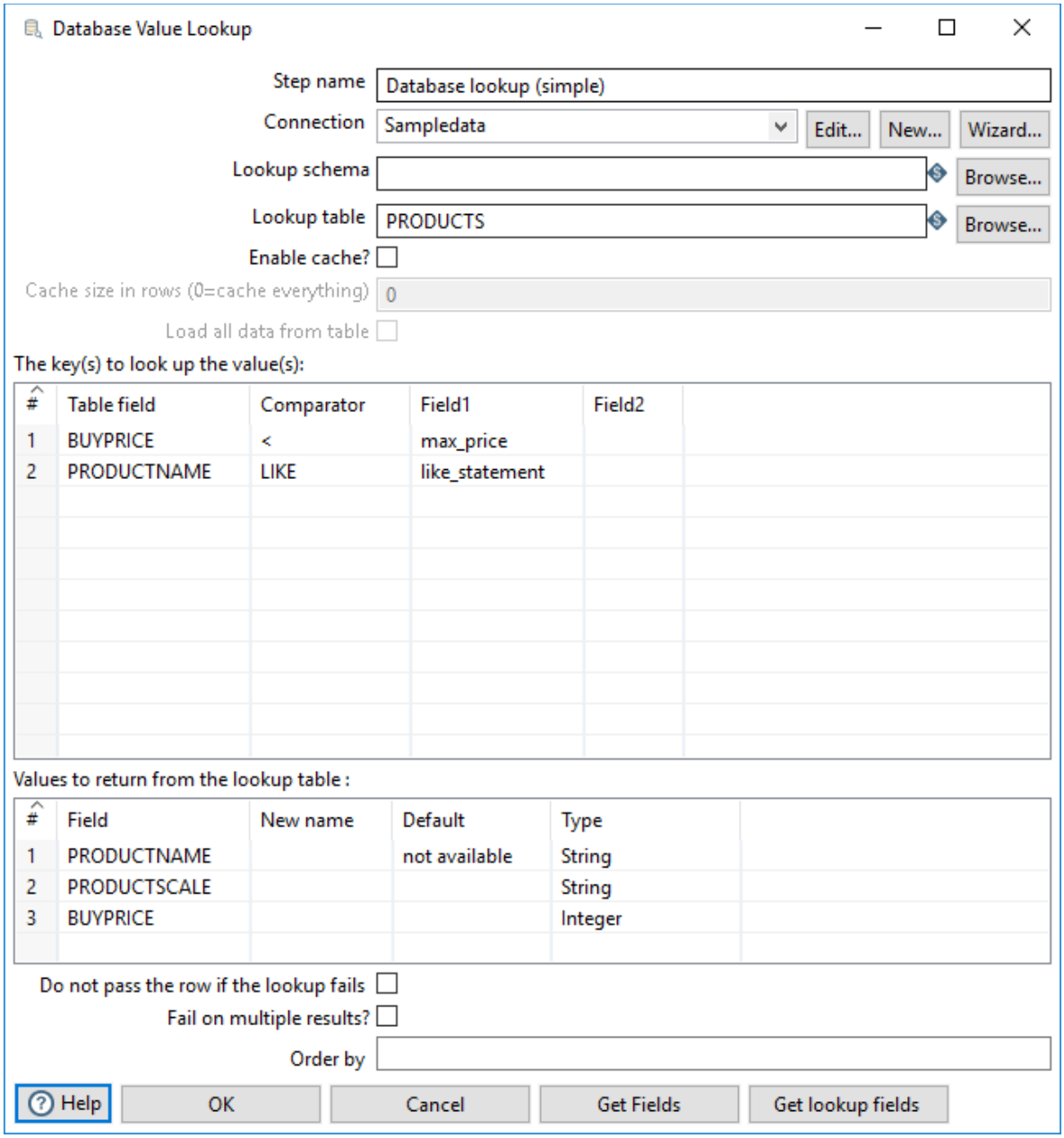


- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

Database Lookup (simple)

The Database lookup step allows you to look up values in a database table. Lookup values are added as new fields onto the stream.

1. Drag the Data Grid step onto the canvas.
2. Open the Data grid properties dialog box.
3. Ensure the following details are configured, as outlined below:



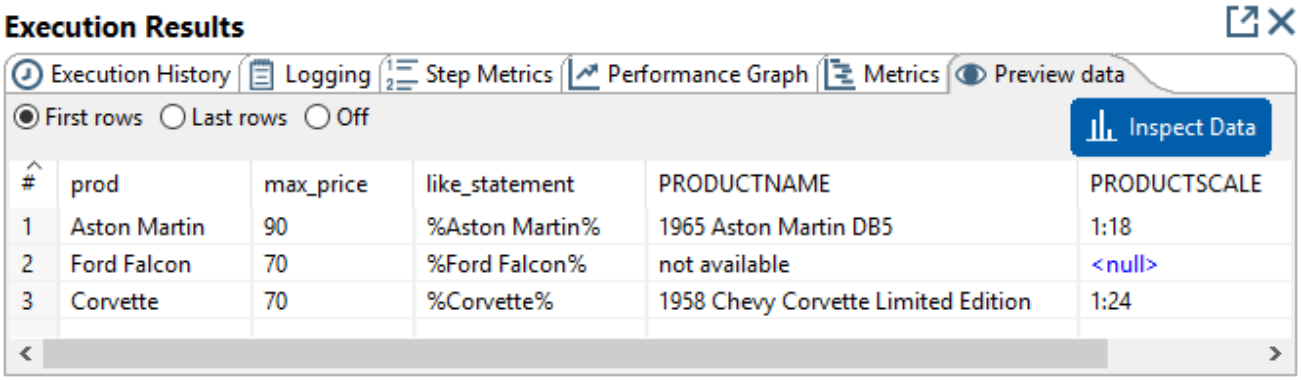
The 'key' fields are where you specify the conditions. Each row in the grid represents a comparison between a column in the table, and a field in your stream, by using one of the provided comparators. In this example:
WHERE PRODUCTNAME LIKE '%Aston Martin%' AND BUYPRICE < 90

WHERE PRODUCTNAME LIKE '%'Ford Falcon%' AND BUYPRICE < 70

WHERE PRODUCTNAME LIKE '%Corvette'%' AND BUYPRICE < 70

The Database lookup step allow us to retrieve any number of columns based on the search criteria. Each database column you enter in the lower grid will become a new field in your dataset. You can rename them (this is particularly useful if you already have a field with the same name) and supply a default value if no record is found in the search.

In the workflow, you added three fields: PRODUCTNAME, PRODUCTSCALE, and BUYPRICE. For values where there's no match for PRODUCTNAME, 'not available' is returned. In the Preview, notice there are no PRODUCTNAMES that match %Ford Falcon% where the max_price is 70.



#	prod	max_price	like_statement	PRODUCTNAME	PRODUCTSCALE
1	Aston Martin	90	%Aston Martin%	1965 Aston Martin DB5	1:18
2	Ford Falcon	70	%Ford Falcon%	not available	<null>
3	Corvette	70	%Corvette%	1958 Chevy Corvette Limited Edition	1:24

Database Lookup (with Error handling)

In this workflow, error handling has been enabled, with a write to log step.

1. To see this in action, disable the Hops to Database Lookup (simple) and Database Lookup (do not pass).
2. The error message is written out in the Logging output.

```
2016/12/10 18:26:48 - Spoon - Launching transformation [GD4-2-1_databse_lookup]...
2016/12/10 18:26:48 - Spoon - Started the transformation execution.
2016/12/10 18:26:48 - GD4-2-1_databse_lookup - Dispatching started for transformation [GD4-2-1_databse_lookup]
2016/12/10 18:26:48 - some conditions.0 - Finished processing (I=0, O=0, R=0, W=3, U=0, E=0)
2016/12/10 18:26:48 - like_statement.0 - Finished processing (I=0, O=0, R=3, W=3, U=0, E=0)
2016/12/10 18:26:48 - Write to log.0 -
2016/12/10 18:26:48 - Write to log.0 - ------------> Linenr 1----------------------------
2016/12/10 18:26:48 - Write to log.0 - no productname match at that price
2016/12/10 18:26:48 - Write to log.0 -
2016/12/10 18:26:48 - Write to log.0 - Ford Falcon
2016/12/10 18:26:48 - Write to log.0 - 70
2016/12/10 18:26:48 - Write to log.0 - %Ford Falcon%
2016/12/10 18:26:48 - Write to log.0 -
2016/12/10 18:26:48 - Write to log.0 - ====================
2016/12/10 18:26:48 - Database lookup (with error handling).0 - Finished processing (I=2, O=0, R=3, W=2, U=0, E=1)
2016/12/10 18:26:48 - Write to log.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2016/12/10 18:26:48 - Spoon - The transformation has finished!!
```

3. Preview the Database Lookup (with error handling) step.

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

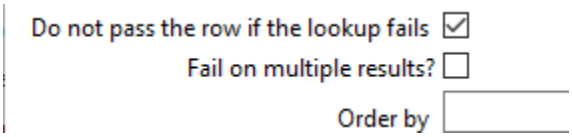
First rows | Last rows | Off — Inspect Data

#	prod	max_price	like_statement	PRODUCTNAME	PRODUCTSCALE
1	Aston Martin	90	%Aston Martin%	1965 Aston Martin DB5	1:18
2	Corvette	70	%Corvette%	1958 Chevy Corvette Limited Edition	1:24

The rows for which the lookup fails, go directly to the stream that captures the error, in this case, the 'Write to log' step.

Database Lookup (do not pass)

1. Selecting the option, 'Do not pass the row if the lookup fails', does what it says..!



2. Preview the Database Lookup (do not pass) step.

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

First rows | Last rows | Off | Inspect Data

#	prod	max_price	like_statement	PRODUCTNAME	PRODUCTSCALE
1	Aston Martin	90	%Aston Martin%	1965 Aston Martin DB5	1:18
2	Corvette	70	%Corvette%	1958 Chevy Corvette Limited Edition	1:24

Taking some action when there are too many results

The Database lookup step is meant to retrieve just one row of the table for each row in your dataset. If the search finds more than one row, the following two things may happen:

1. If you check the Fail on multiple results? option, the rows for which the lookup retrieves more than one row will cause the step to fail. In that case, in the Logging tab window, you will see an error similar to the following:

   ... - Database lookup (fail on multiple res.).0 – ERROR...

   Because of an error, this step can't continue:

   ... - Database lookup (fail on multiple res.).0 – ERROR... :

   Only 1 row was expected as a result of a lookup, and at least 2 were found!

   Then you can decide whether you want to leave the transformation or capture the error.

2. If you don't check the Fail on multiple results? option, the step will return the first row it encounters. You can decide which one to return by specifying the order. You do that by typing an order clause in the Order by textbox. In the Sampledata database, there are three products that meet the conditions for the Corvette row. If, for Order by, you type PRODUCTSCALE DESC, PRODUCTNAME, then you will get 1958 Chevy Corvette Limited Edition, which is the first product after ordering the three found products by the specified criterion.

If, instead of taking some of those actions, you realize that you need all the resulting rows, you should take another approach—replace the Database lookup step with a Database join or a Dynamic SQL row step.

Compare this with the Database Join GD 4-3-3

As the database join is a full outer, all the records are returned from the database table, rather than just return a single lookup reference value.

There's often a careful balancing act to perform with ETL workflows. For example, consider schema changes: new data sources need to be added, and the data model needs to be updated. In the manual route, you would need to go back to the drawing board, make the necessary changes and then run a series of test to make sure you haven't broken anything; in larger data teams or deployments, multiple people need to be intimately familiar with the code whenever a simple modification is needed.

On the other hand, a visual ETL layer makes the process much more transparent and easy to modify, as well as troubleshoot in case things go wrong. And more importantly: every change is clearly documented, immediately noticeable and can be retraced – without having to spend hours or days poring over code to locate a change made by one of the developers, who in the meantime has gone on his annual vacation.

Pentaho data Integration has several steps / job entries that help you automate the workflow. One of the most flexible, is the job entry: Execute a shell script:

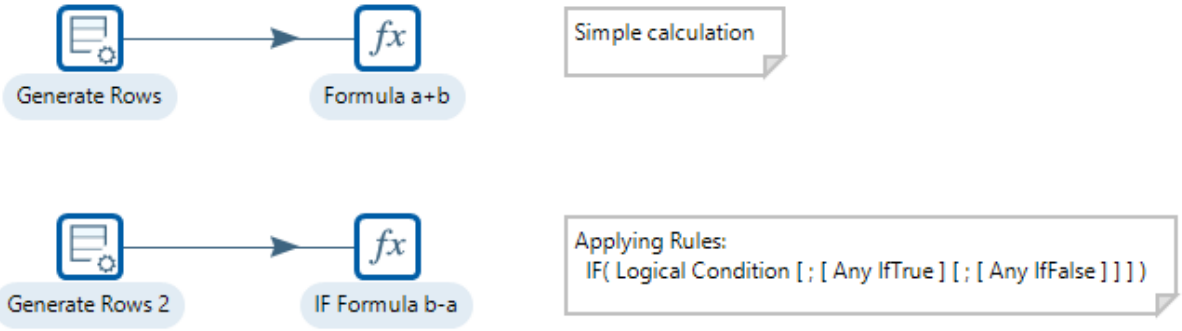
Use the **Shell** job entry to execute a shell script on the host where the job is running. For example, suppose you have a program that reads five data tables and creates a file in a specified format. You know the program works. Shell allows you to do portions of your work in Pentaho Data Integration but reuse the program that reads the data tables as needed.

The Shell job entry is platform agnostic; you can use a batch file, UNIX, and so on. When you use a Shell job entry, Pentaho Data Integration makes a Java call to execute a program in a specified location. The return status is provided by the operating system call. For example, in batch scripting a return value of 1 indicates that the script was successful; a return value of 0 (zero) indicates that it was unsuccessful. You can pass command line arguments and set up logging for the Shell job entry.

Shell scripts can output text to the console window. This output will be transferred to the Kettle logging system. Doing this no longer blocks the shell script.

Introduction	Demonstration to illustrate the Formula step.
Objectives	In this guided demonstration, you will: • Configure the following steps: ○ Formula

Transformation



This demonstration highlights the use of the Formula step.

Generate Rows

Generate rows outputs a specified number of rows. By default, the rows are empty; however, they can contain several static fields. This step is used primarily for testing purposes. It may be useful for generating a fixed number of rows, for example, you want exactly 12 rows for 12 months. Sometimes you may use Generate Rows to generate one row that is an initiating point for your transformation. For example, you might generate one row that contains two or three field values that you might use to parameterize your SQL and then generate the real rows.

Used to generate some testing data:

a = 1

b = 2

booking_type = R

Formula

The Formula step can calculate Formula Expressions within a data stream. It can be used to create simple calculations like [A]+[B] or more complex business logic with a lot of nested if / then logic.

Further details can be found at:

`http://wiki.pentaho.com/display/Reporting/Formula+Expressions`

Oasis OpenFormula syntax (also used in Calc, OpenOffice)

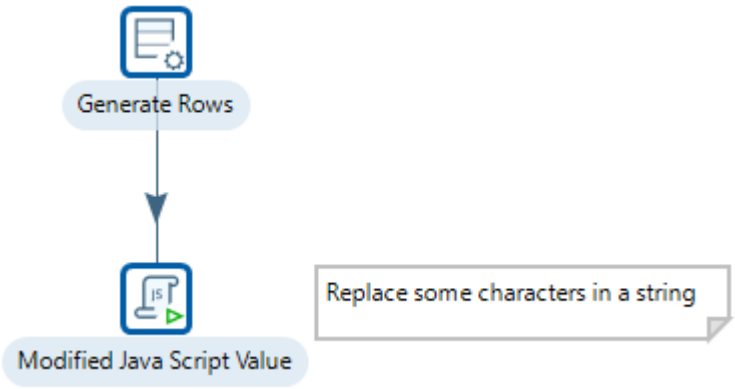
`http://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2-part2.html`

Workflow 1

Simple formula: c = [a] + [b]

Workflow 2

Logic: c = IF([booking_type]="R";[b]-[a])

Introduction	Demonstration to illustrate the MJV step.
Objectives	In this guided demonstration, you will: • Configure the following steps: ○ Modified JavaScript Value
Transformation	

This demonstration highlights the use of the Modified JavaScript Value step. Uses the Replace () to replace a few illegal characters.

Generate Rows

Generate rows outputs a specified number of rows. By default, the rows are empty; however, they can contain a number of static fields. This step is used primarily for testing purposes. It may be useful for generating a fixed number of rows, for example, you want exactly 12 rows for 12 months. Sometimes you may use Generate Rows to generate one row that is an initiating point for your transformation. For example, you might generate one row that contains two or three field values that you might use to parameterize your SQL and then generate the real rows.

field: This is a String \ with / some weird characters - in it!

fieldsame: This is a String // with - some weird characters - in it!

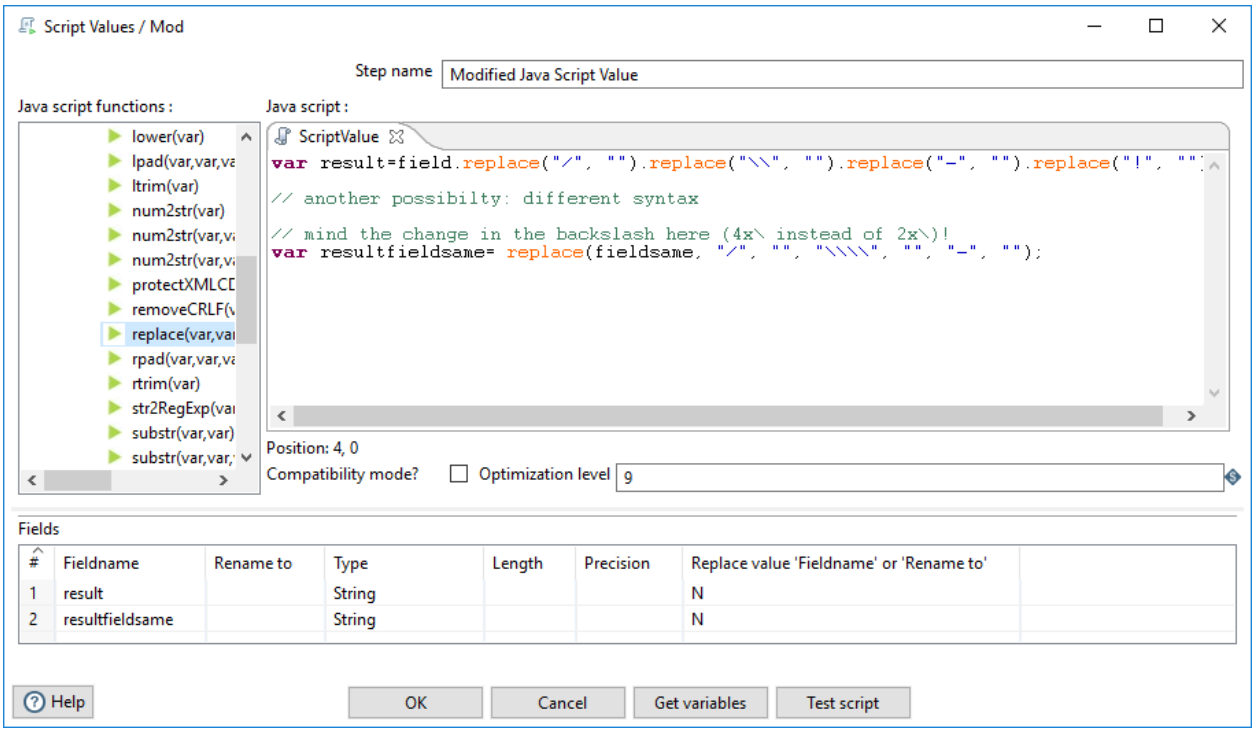
Modified JavaScript Value

This is a modified version of the 'JavaScript Values' step that provides better performance and an easier, expression based user interface for building JavaScript expressions. This step also allows you to create multiple scripts for each step.

Java script functions

This section provides a tree view of your available scripts, functions, input fields and output fields.

- **Transformation Scripts:** displays a list of scripts you have created in this step
- **Transformation Constants:** a list of pre-defined, static constants including SKIP_TRANSFORMATION, ERROR_TRANSFORMATION, and CONTINUE_TRANSFORMATION
- **Transformation Functions:** contains a variety of String, Numeric, Date, Logic and specialized functions you can use to create your script. To add a function to your script, simply double-click on the function or drag it to the location in your script that you wish to insert it.
- **Input Fields:** a list of inputs coming into the step. Double-click or use drag and drop to insert the field into your script.
- **Output Fields:** a list of outputs for the step.



Code:

```
var result=field.replace("/", "").replace("\\", "").replace("-", "").replace("!", "");

// another possibility: different syntax

// mind the change in the backslash here (4x\ instead of 2x\)!
var resultfieldsame= replace(fieldsame, "/", "", "\\\\", "", "-", "");
```

Summary

Topics covered in this section:

- Stream Operations
  - Merge rows
  - Merge rows (diff)
- Lookups
  - Database Lookups
- Joins
  - Join rows
  - Merge Join
  - Database Join
- Scripting
  - Formula
  - Modified JavaScript Value

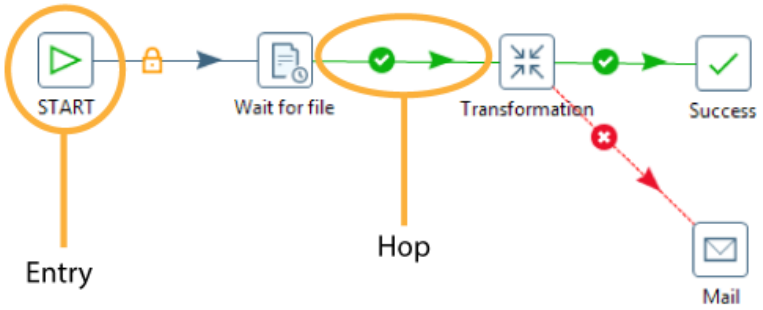
Pentaho Data Integration has several steps that enable you to enrich data with the following types of information:

- Geographic: such as postcode, county name, longitude and latitude, and political district
- Behavioural: including purchases, credit risk and preferred communication channels
- Demographic: such as income, marital status, education, age and number of children
- Psychographic: ranging from hobbies and interests to political affiliation
- Census: household and community data

Jobs are workflows whose role is to orchestrate the execution of a set of tasks: they generally synchronize and prioritize the execution of tasks and give an order of execution based on the success or failure of the execution of the current task. These tasks are basic tasks that either prepare the execution environment for other tasks that are next in the execution workflow or that manage the artefacts produced by tasks that are preceding them in the execution workflow. For example, we have tasks that let us manipulate the files and directories in the local filesystem, tasks that move files between remote servers through FTP or SSH, and tasks that check the availability of a table or the content of a table. Any job can call other jobs or transformations to design more complex processes. Therefore, generally speaking, jobs orchestrate the execution of jobs and transformations into large ETL processes.

Carte is a simple web server that allows you to run transformations and jobs remotely. It receives XML (using a small servlet) that contains the transformation to run and the execution configuration. It allows you to remotely monitor, start and stop the transformations and jobs that run on the Carte server.

You can set up an individual instance of Carte to operate as a standalone execution engine for a job or transformation. In Spoon, you can define one or more Carte servers and send jobs and transformations to them. If you want to improve PDI performance for resource-intensive transformations and jobs, use Carte cluster.



In most ETL tasks you need to be able to perform maintenance tasks, orchestrate the execution of the transformations, and handle errors and retries. These tasks are handled by Jobs.

A job consists of one or more job entry that are executed in a specific order. The order of the execution is determined by the job hops between the job entries as well as the executions themselves. Job entries differ in several ways:

- You can create shadow copies of a job entry. This allows to place the same job entry in a job on multiple locations.
- A job entry passes a results object between job entries. This means that once a job entry has been completed all rows are transferred at once, rather than in a streaming fashion.
- Job entries are executed in a certain sequence (except if set to parallel execution)

Job Hops

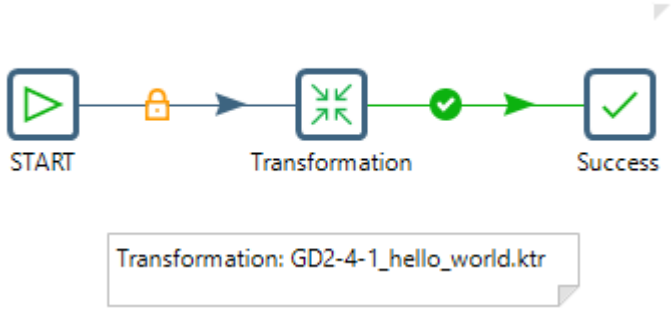
Besides the execution order, a hop also specifies the condition on which the next job entry will be executed. You can specify the Evaluation mode by right clicking on the job hop. A job hop is just a flow of control. Hops link to job entries and, based on the results of the previous job entry, determine what happens next.

Guided Demo 5-1-1: Hello World (Jobs)

Introduction	In this guided demonstration, we will use Spoon to create a new Job that utilizes: GD5-1-1_hello_world.ktr. The concepts learnt, help to build the foundation necessary for creating any Job.

Objectives	In this guided demonstration, you will:
	• Learn to create a new Job.
	• Add steps and hops.
	• Configure the following steps:
	  ○ Generate Rows
	  ○ Dummy

Transformation	A simple Job that runs the Hello World Transformation.



To create a new transformation:

1. In Spoon, click File > New > Job.

Any one of these actions opens a new Transformation tab for you to begin designing your transformation.

- By clicking New, then Job
- By using the CTRL-ALT-N hot key

START

START defines the starting point for job execution. Every job must have one (and only one) Start. Unconditional job hops only are available from a Start job entry.

The start job entry settings contain basic scheduling functionality; however, scheduling is not persistent and is only available while the device is running.

The Data Integration Server provides a more robust option for scheduling execution of jobs and transformations and is the preferred alternative to scheduling using the Start step. If you want the job to run like a daemon process, however, enable **Repeat** in the job settings dialog box.

Note: The basic scheduling functionality and the repeat option are only functional within the main job and not within a sub job.

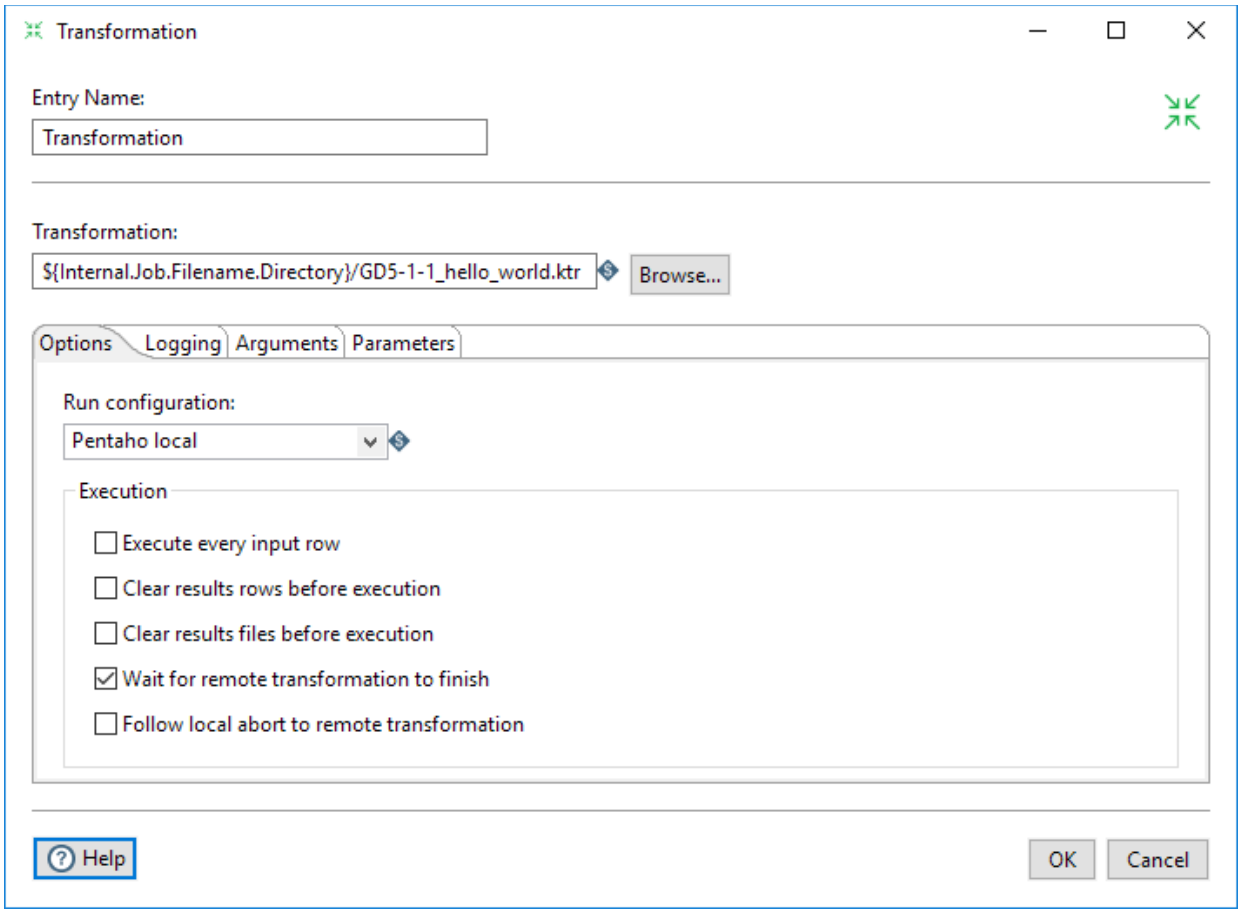
1. Drag the 'START' job entry onto the canvas.

Transformation

The Transformation job entry is used to execute a previously defined transformation.

For ease of use, it is also possible to create a new transformation within the dialog, pressing the *New Transformation* button.

1. Drag the 'Transformation' job entry onto the canvas.
2. Double-click on the step, and configure the following properties:



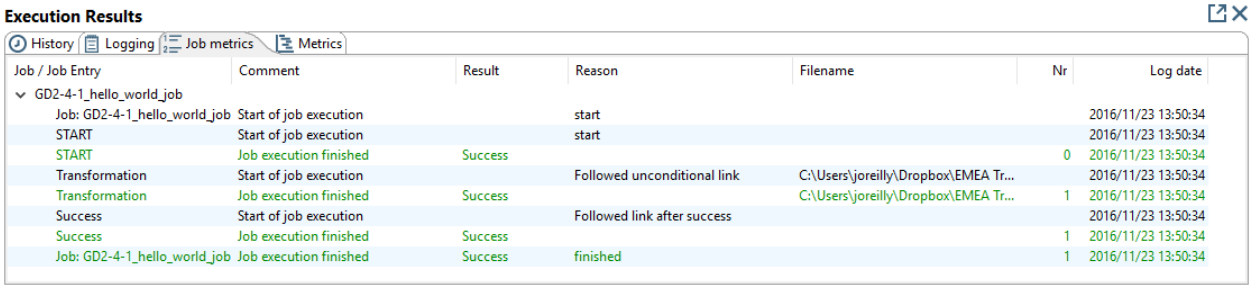
Success

This step clears any error state encountered in a job and forces it to a success state.

1. Drag the 'Success' job entry onto the canvas.

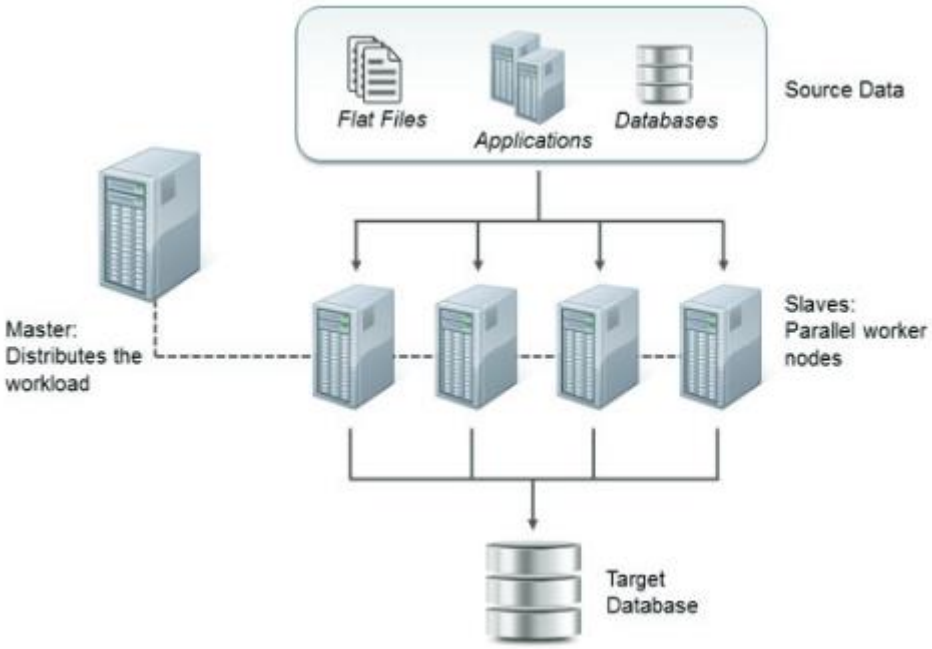
RUN the Transformation

1. Click the Run button in the Canvas Toolbar.
2. Click on the Job Metrics tab.



Scalability - Static and Dynamic Carte Clusters

If you want to speed the processing of your transformations, consider setting up a PDI Carte cluster. A PDI Carte cluster consists of two or more Carte slave servers and a Carte master server. When you run a transformation, the different parts of it are distributed across Carte slave server nodes for processing, while the Carte master server node tracks the progress.



Slave Server

A slave server is essentially a small embedded webserver. We use this web-server to control the slave server. The slave server is started with the Carte program found in your PDI distribution.

The arguments used to start a slave server with Carte are discussed elsewhere but the minimum is always an IP-address or hostname and a HTTP port to communicate. For example:

```
sh carte.sh localhost 8000
carte.bat   localhost 8000
```

The slave server metadata can be entered in the Spoon GUI by basically giving your Carte instance a name and entering the same data.

There are two types of Carte clusters:

Static Carte cluster has a fixed schema that specifies one master node and two or more slave nodes. In a static cluster, you specify the nodes in a cluster at design-time, *before* you run the transformation or job.

A Dynamic Carte cluster has a schema that specifies one master node and a varying number of slave nodes. Unlike a static cluster, slave nodes are not known until runtime. Instead, you register the slave nodes, then at runtime, PDI monitors the slave nodes every 30 seconds to see if it is available to perform transformation and job processing tasks.

Static clusters are a good choice for smaller environments where you don't have a lot of machines (virtual or real) to use for PDI transformations. Dynamic clusters work well if nodes are added or removed often, such as

in a cloud computing environment. Dynamic clustering is also more appropriate in environments where transformation performance is extremely important, or if there can potentially be multiple concurrent transformation executions.

Cluster schema

A cluster schema is essentially a collection of slave servers. In each schema, you need to pick at least one slave server that we will call the Master slave server or master. The master is also just a carte instance but it takes care of all sort of management tasks across the cluster schema.

In the Spoon GUI, you can enter this metadata as well once you started a couple of slave servers.

Configure a Static Carte Cluster

Follow the directions below to set up static Carte slave servers.

1. Copy over any required JDBC drivers and PDI plugins from your development instances of PDI to the Carte instances.
2. Run the Carte script with an IP address, hostname, or domain name of this server, and the port number you want it to be available on:

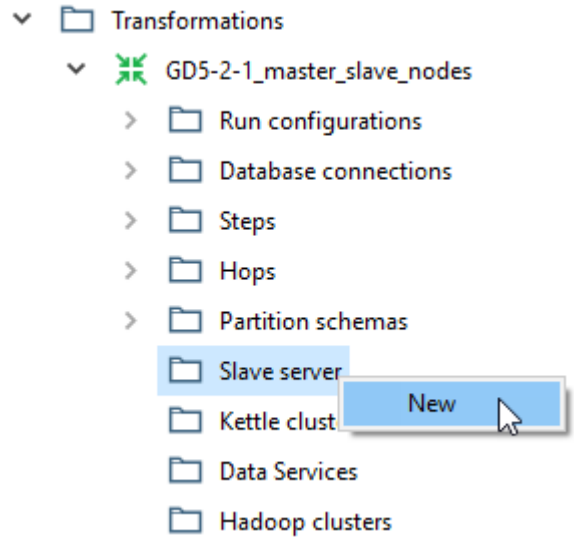
   ```
   sh carte.sh 192.168.77 8000
   carte.bat   192.168.77 8000
   ```

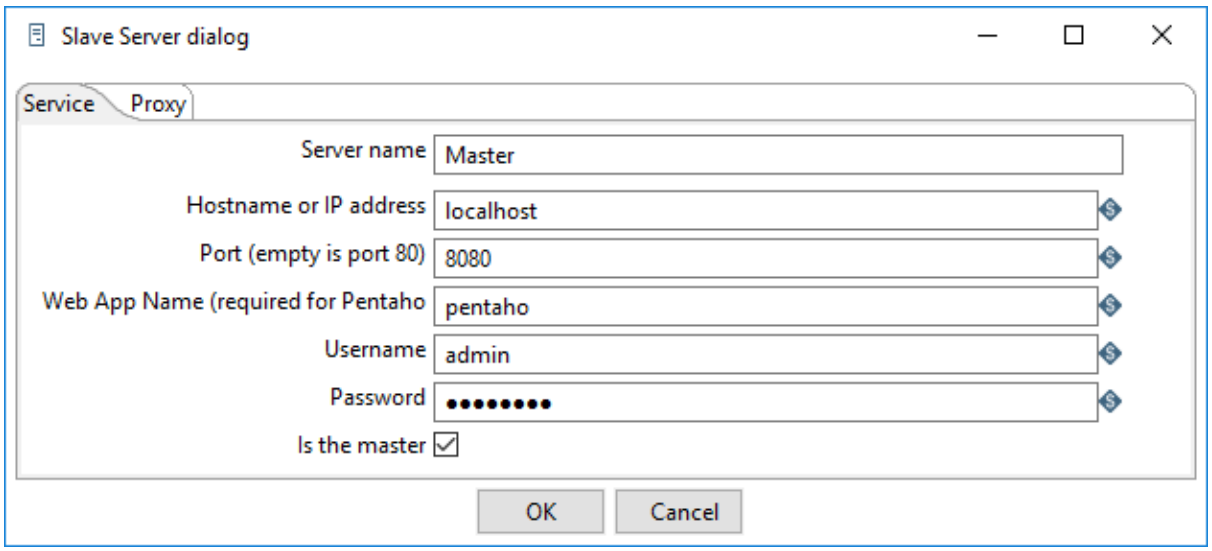
3. If you will be executing content stored in a Pentaho Repository, copy the **repositories.xml** file from the .kettle directory on your workstation to the same location on your Carte slave. Without this file, the Carte slave will be unable to connect to the Pentaho Repository to retrieve content.
4. Ensure that the Carte service is running as intended, accessible from your primary PDI development machines, and that it can run your jobs and transformations.
5. To start this slave server every time the operating system boots, create a startup or init script to run Carte at boot time with the same options you tested with.

Introduction	Pentaho Data Integration provides advanced clustering and partitioning capabilities that allow organizations to scale out their data integration deployments.
Objectives	In this guided demonstration, you will: • Configure Master & Slave Nodes • Configure a Cluster Schema

Configure PDI to work with Carte Master / Slave nodes.

1. Open a transformation GD5-2-1 – Master-Slave Nodes and save it to the Repository.
2. In the Explorer View in Spoon, select the Slave tab.
3. Select the New button. The Slave Server dialog window appears.





4. In the Slave Server dialog window, enter the appropriate connection information for the Pentaho (or Carte) slave server.

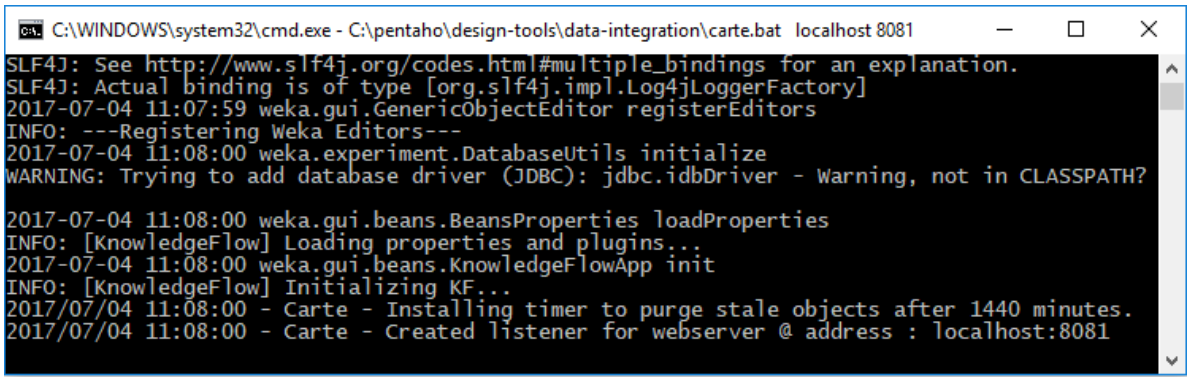
Option	Description
Server name	The name of the slave server.
Hostname or IP address	The address of the device to be used as a slave.
Port (empty is port 80)	Defines the port you are for communicating with the remote server. If you leave the port blank, 80 is used.
Web App Name (required for Pentaho Server)	Leave this blank if you are setting up a Carte server. This field is used for connecting to the Pentaho server.
User name	Enter the user name for accessing the remote server.
Password	Enter the password for accessing the remote server. (password)
Is the master	Enables this server as the master server in any clustered executions of the transformation.

Below are the proxy tab options:

Option	Description
Proxy server hostname	Sets the host name for the proxy server you are using.
The proxy server port	Sets the port number used for communicating with the proxy.
Ignore proxy for hosts: regexp | separated	Specify the server(s) for which the proxy should not be active. This option supports specifying multiple servers using regular expressions. You can also add multiple servers and expressions separated by the ' | ' character.

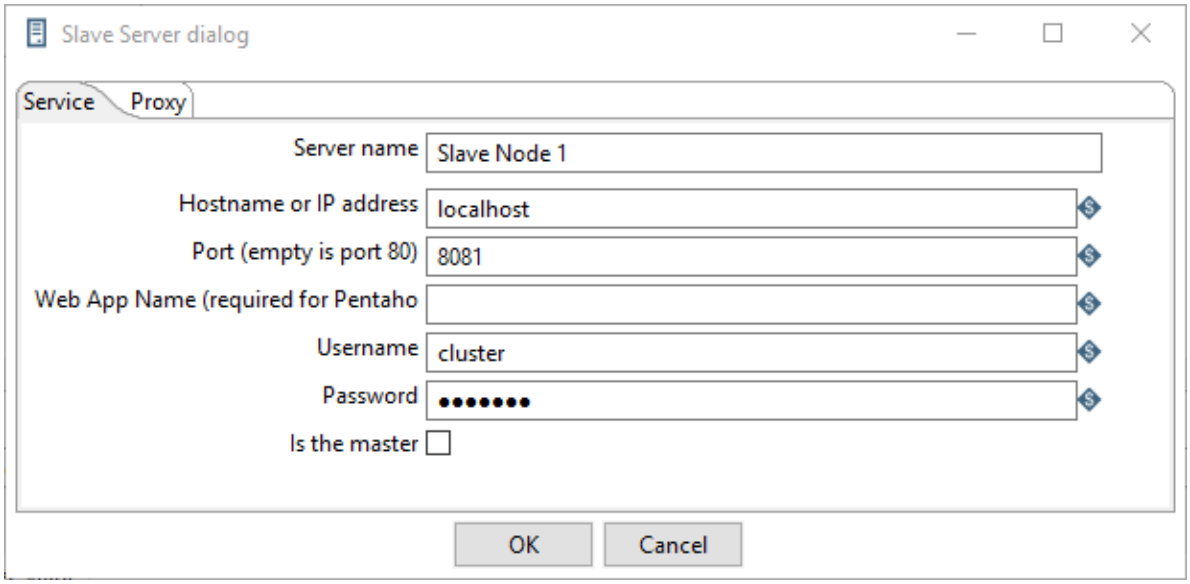
Configure Slave Node:

1. Open a CMD window and navigate to the data-integration directory.
   `C:\Pentaho\design-tools\data-integration`
2. Execute the following command to start a carte instance on port: 8081
   `carte.bat localhost 8081`



- Or.. double-click on the start slave node 1 8081.bat file
- Don't close the window..!

3. In Spoon, repeat the workflow in the Design tab, to declare the Slave Node 1:

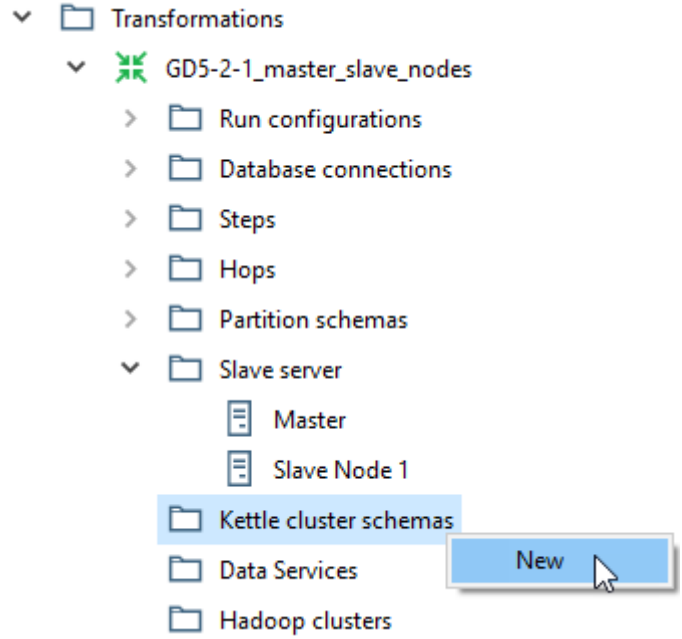


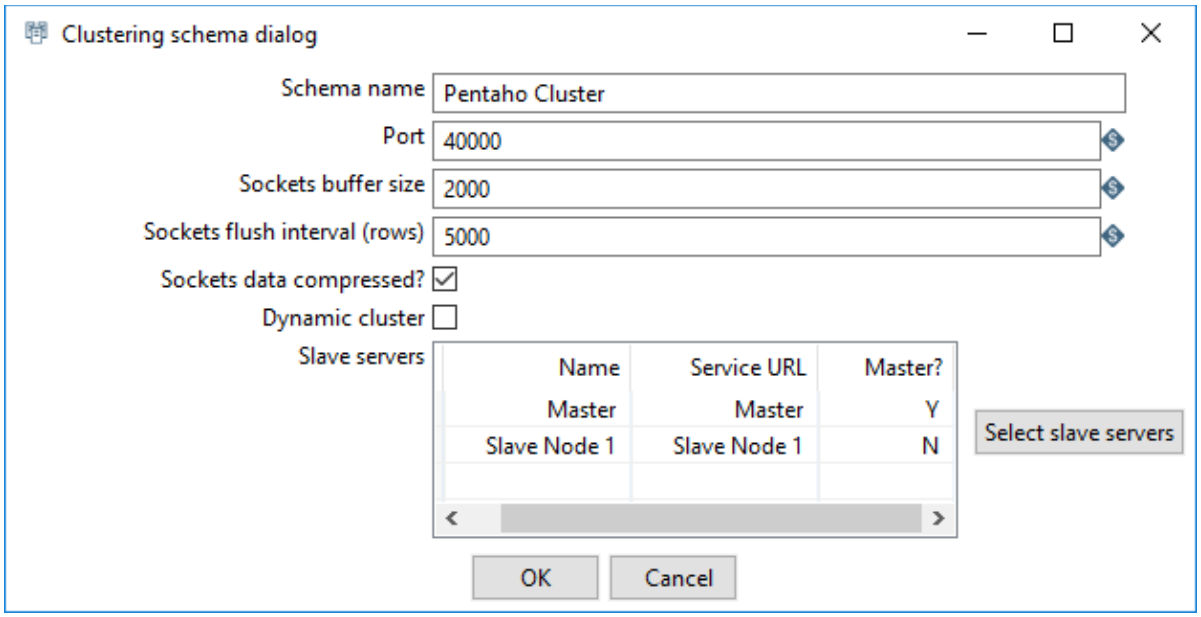
- **Password:** cluster

Define a Cluster Schema in Spoon

Clustering allows transformations and transformation steps to be executed in parallel on more than one Carte server. The clustering schema defines which slave servers you want to assign to the cluster and a variety of clustered execution options.

1. Begin by selecting the Kettle cluster schemas node in the Spoon Explorer View. Right-click and select New to open the Clustering Schema dialog box.



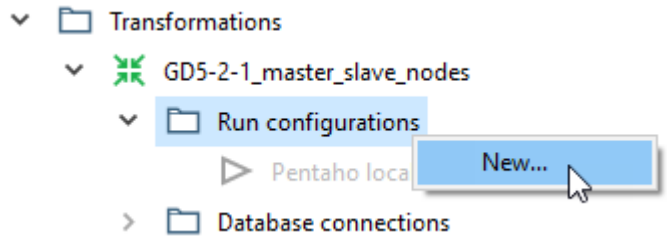


Option	Description
Schema name	The name of the clustering schema
Port	Specify the port from which to start numbering ports for the slave servers. Each additional clustered step executing on a slave server will consume an additional port. Note: To avoid networking problems, make sure no other networking protocols are in the same range.
Sockets buffer size	The internal buffer size to use
Sockets flush interval rows	The number of rows after which the internal buffer is sent completely over the network and emptied.
Sockets data compressed?	When enabled, all data is compressed using the Gzip compression algorithm to minimize network traffic
Dynamic cluster	If checked, a master Carte server will perform failover operations, and you must define the master as a slave server in the field below. If unchecked, Spoon will act as the master server, and you must define the available Carte slaves in the field below.
Slave Servers	A list of the servers to be used in the cluster. You must have one master server and any number of slave servers. To add servers to the cluster, click **Select slave servers** to select from the list of available slave servers.

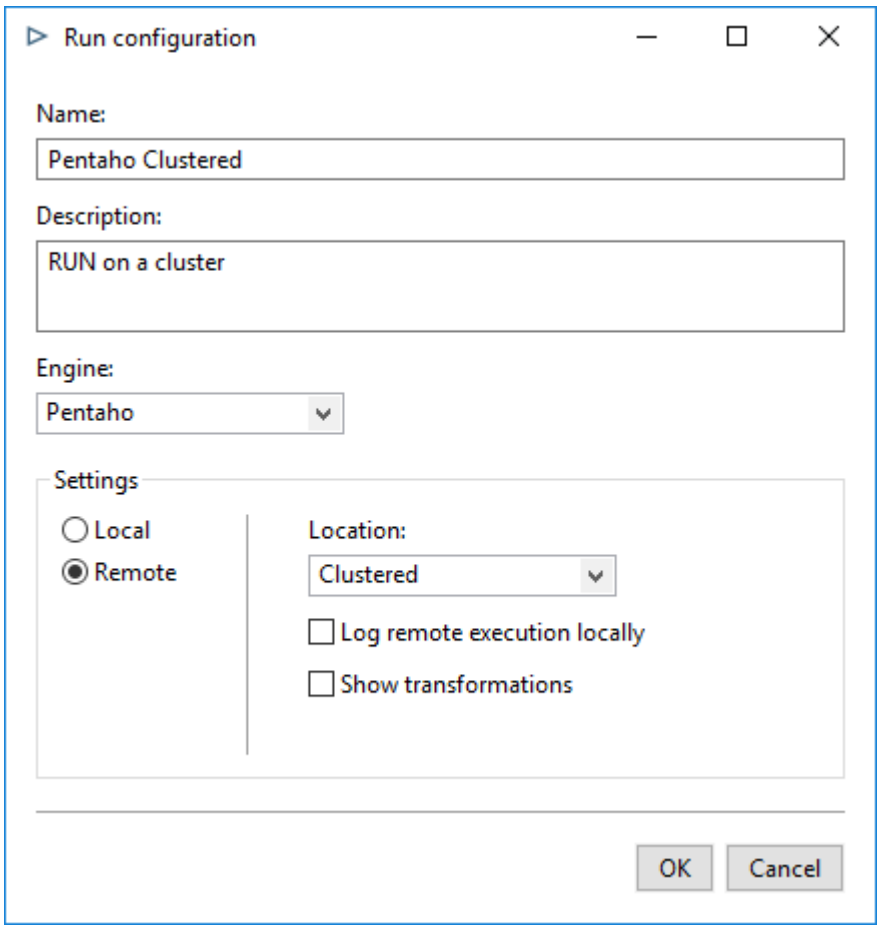
RUN Configurations

Some ETL activities are lightweight, such as loading in a small text file to write out to a database or filtering a few rows to trim down your results. For these activities, you can run your transformation locally using the default Pentaho engine. Some ETL activities are more demanding, containing many steps calling other steps or a network of transformation modules. For these activities, you can set up a separate Pentaho Server dedicated for running transformations using the Pentaho engine. Other ETL activities involve large amounts of data on network clusters requiring greater scalability and reduced execution times. For these activities, you can run your transformation using the Spark engine in a Hadoop cluster.

1. To create a new run configuration, right-click on the **Run Configurations** folder and select **New**, as shown in the folder structure below:



2. Enter the following configuration details, ensuring that you select the Pentaho (KETTLE) engine.



Pentaho local is the default run configuration. It runs transformations with the Pentaho engine on your local machine. You cannot edit this default configuration.

Introduction

There are a few ways that you can monitor step performance in PDI. Two tools are particularly helpful: the Sniff Test tool and the Monitoring tab.

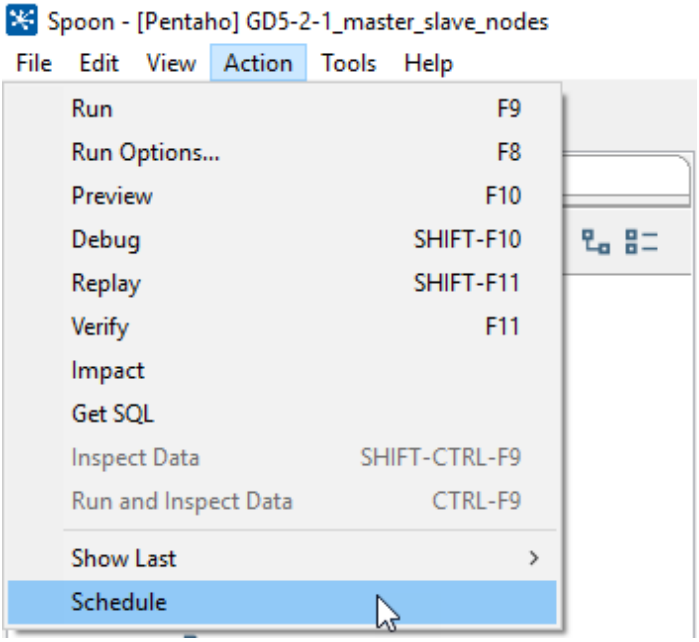
Objectives

In this guided demonstration, you will:

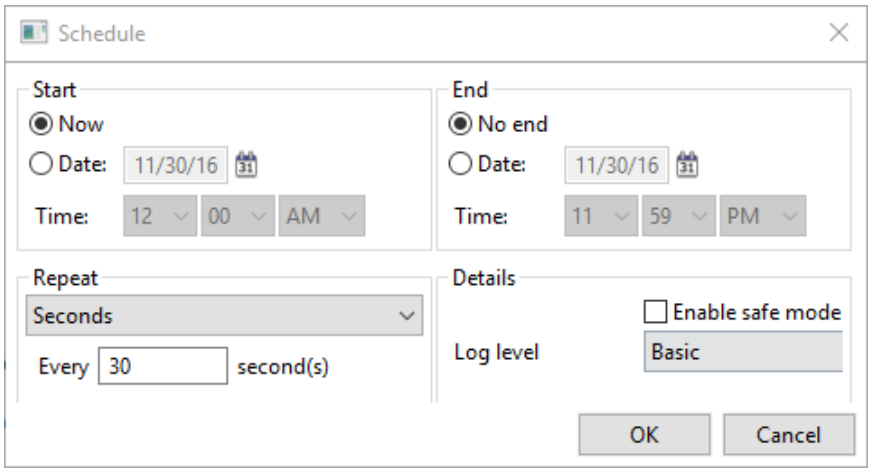
- Schedule a Transformation
- Monitor Transformation

You can schedule jobs and transformations to execute automatically on a recurring basis by following the directions below.

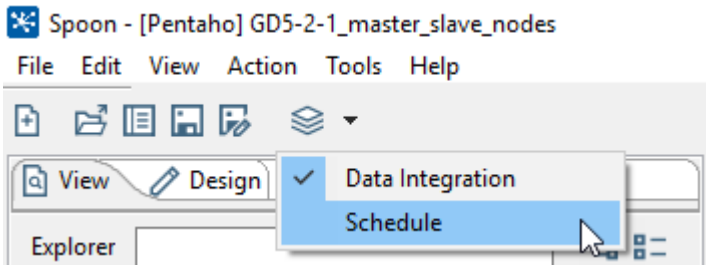
1. Open a transformation GD5-2-1 – Master – Slave Node from the Repository.
2. In the Action menu and select Schedule.



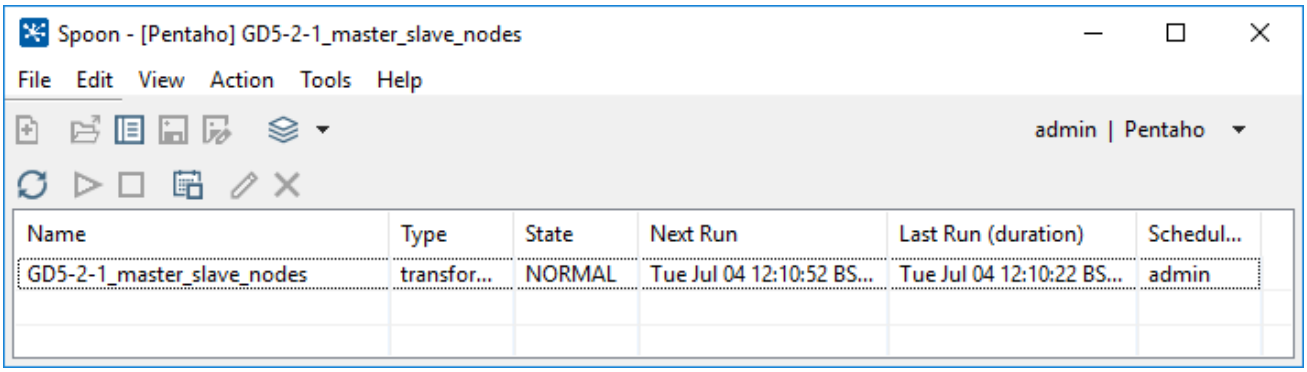
3. Set the schedule to Run the Transformation every 30 seconds.



4. In the Schedule a Transformation dialog box, enter the date and time that you want the schedule to begin in the Start area, or click the calendar icon to display the calendar.
   - To run the transformation immediately, enable the Now radio button.
   - Set up the End date and time. If applicable, enable the No end radio button or click on the calendar and input the date and time to end the transformation.
   - If applicable, set up a recurrence under Repeat.
   - End date and time are disabled unless you select a recurrence. From the list of schedule options select the choice that is most appropriate: Run Once, Seconds, Minutes, Hourly, Daily, Weekly, Monthly, Yearly.
5. Make sure you set parameters, arguments and variables, if available. Click OK.
6. In the Spoon button bar, click the Schedule perspective.



7. From the Schedule perspective, you can refresh, start, pause, stop and delete a transformation or job using the buttons on the upper left corner of the page.

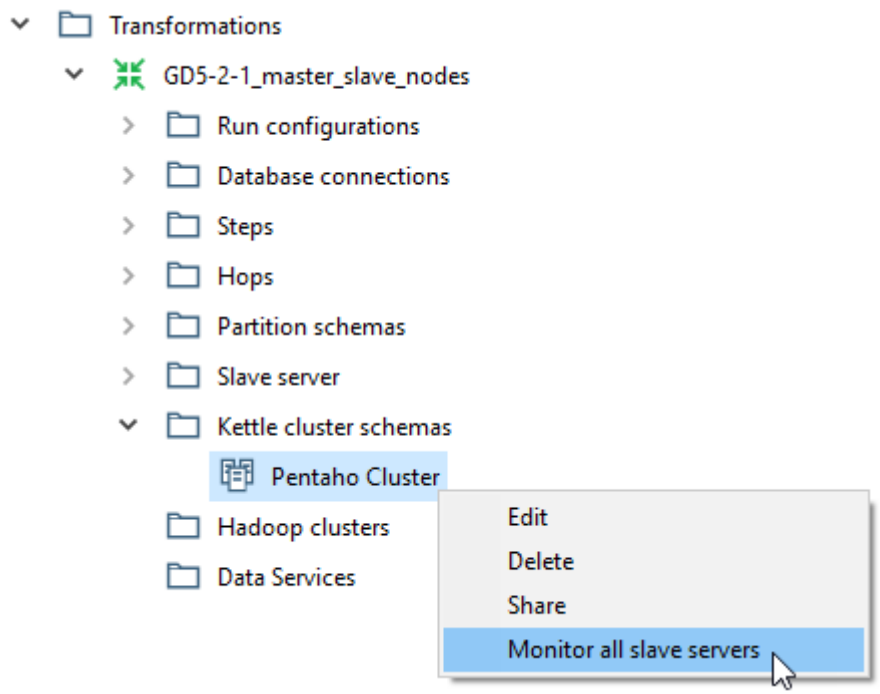


8. Switch back to the Data integration perspective.

To Monitor

In the view tab:

1. Expand the Kettle cluster schemas folder.
2. Right mouse click, and select; Monitor all slave servers



3. Check that each node is up and running. No http:// error messages in any of the monitor panels.
4. Check that you can connect to the nodes:

   Master Node:

   http://localhost:8080/pentaho/kettle/status

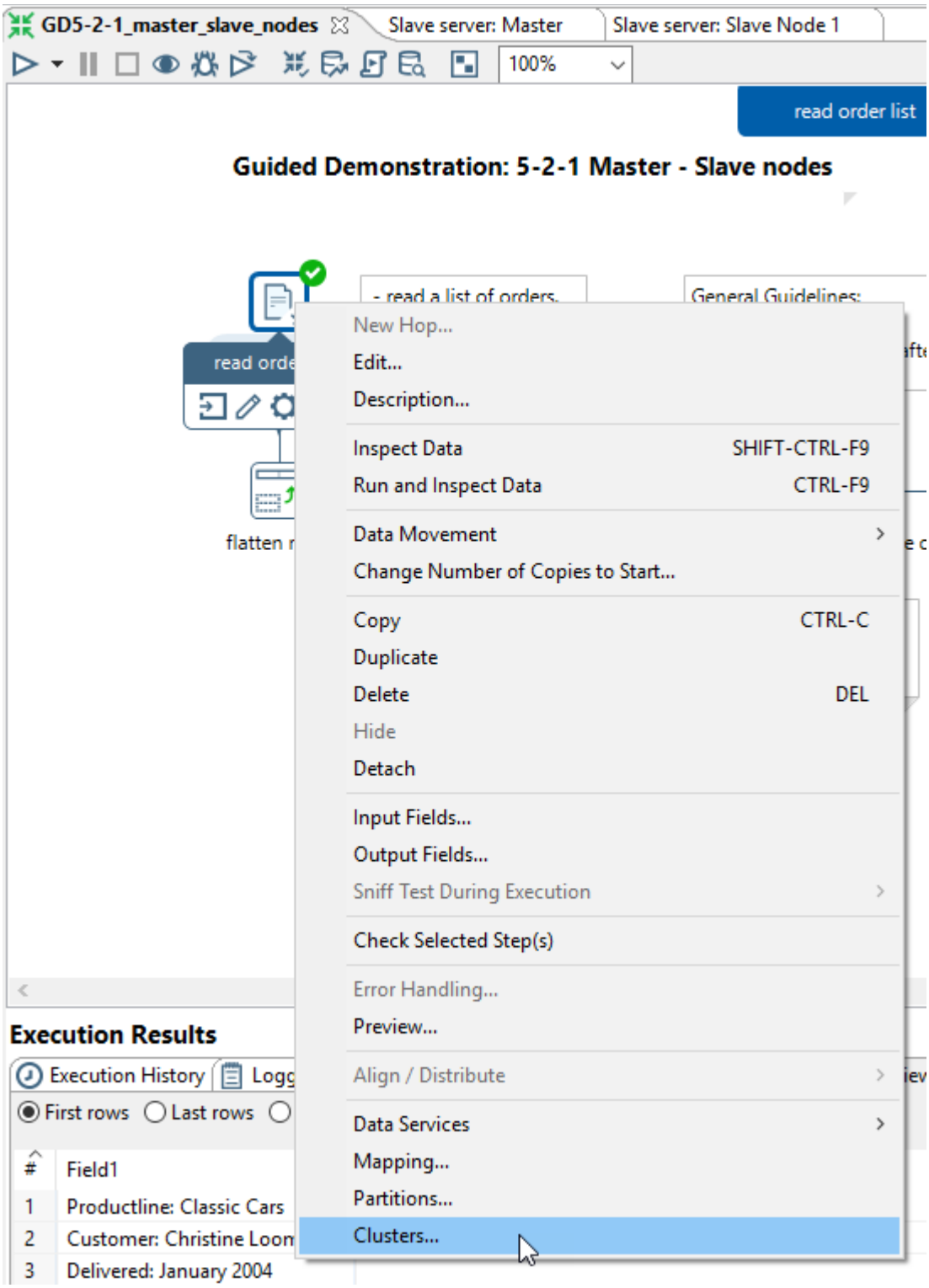
   Username: admin

   Password: password

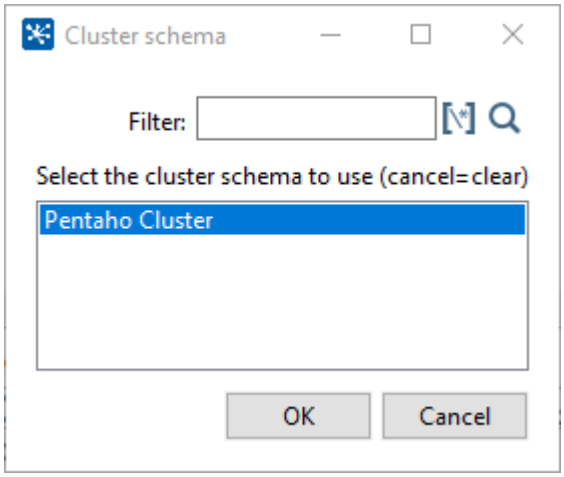
Cluster the read order list step

To Run the *read order list* step in the cluster:

1. Highlight the step and right-mouse click, and from the drop-down options
2. Select the option: 'Custers'



3. Select: Pentaho Cluster

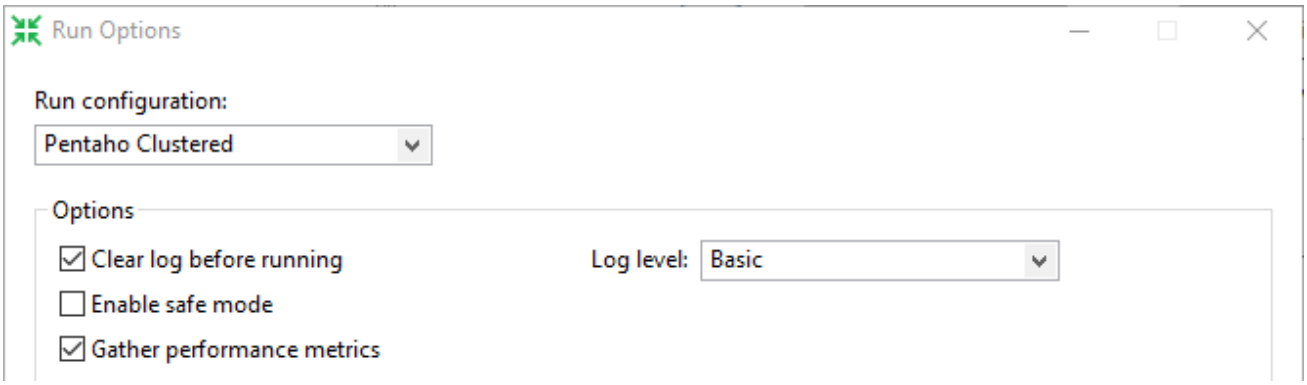


Notice a Cx1 appears by the step. This indicates that the Cluster has 1 slave node.

4. You're now ready to RUN the Transformation.

RUN Transformation in a Cluster

To run a transformation on a cluster, access the Run Options window and select Pentaho Clustered'. You can access the Run Options window through the context menu next to the Run icon in the toolbar or by pressing F8.



Refresh the kettle Transformation status.

The 'read order list' step has been enabled to RUN in the Cluster, i.e. on Slave Node 1. When you examine the Monitoring tabs, you will notice that Slave 1 executed just the 'read order list' step, while the other steps were executed on the Master node.

The Master node is also responsible for collating the result set.

Logging

PDI is configured to provide helpful log messages to help provide understanding in how a job or transformation is running. Logging can be configured to provide minimal logging information, just to know whether a job or transformation failed or was successful, or detailed in providing errors or warnings such as network issues or mis-configurations.

- Nothing: Don't show any output
- Error: Only show errors
- Minimal: Only use minimal logging
- Basic: This is the default basic logging level
- Detailed: Give detailed logging output
- Debug: For debugging purposes, very detailed output.
- Row level: Logging at a row level, this can generate a lot of data.

When executing a job or transformation from within the Spoon development environment, a "Logging" tab is available, showing any log messages that have been generated. Any error messages are shown with red text, to easily identify the cause of any errors in the file.

The next stage is to implement database logging.

Introduction

In Pentaho v7.1 the implementation of logging to a database was standardized. The kettle.properties file includes variables for each of the logging options for Transformations and Jobs.

A logging schema, pentaho_dilogs has been created in the Hibernate PostgreSql database.

This demonstration takes you through the workflow for implementing logging prior to Pentaho v7.1

Objectives

In this guided demonstration, you will:

- Configure kettle.properties file for logging
- Create Logging Database Tables in PostgreSQL Hibernate database.
- Examine Logging tables

The logging functionality in Data Integration enables you to more easily troubleshoot complex errors and failures, and measure performance. Here is how to turn on logging in Data Integration.

1. Edit the kettle.properties file to globally set the logging properties. A template can be found at:

   `C:\Pentaho Training\DI 1000\Module 4 - Logging\Lesson 1 - Logging\Demo 4-1-1 Logging\kettle.properties`
2. Rename your kettle.properties file to kettle.properties.original
3. Copy the kettle.properties to:

   `C:\Users\<user>\.kettle` folder
4. Test that you can access the variables.

Create logging tables in PostgreSQL

For this demonstration, you will require pgAdmin 3 or 4 to create the required schema and database in PostgreSQL. This is an open source administration and management tool for PostgreSQL database. Alternatively, the Logging database can be installed on any standard relational database.

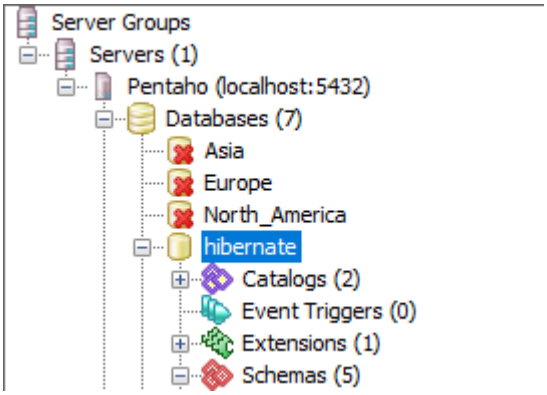
pgAdmin III is installed with a default installation of Pentaho.

To start pgAdmin III

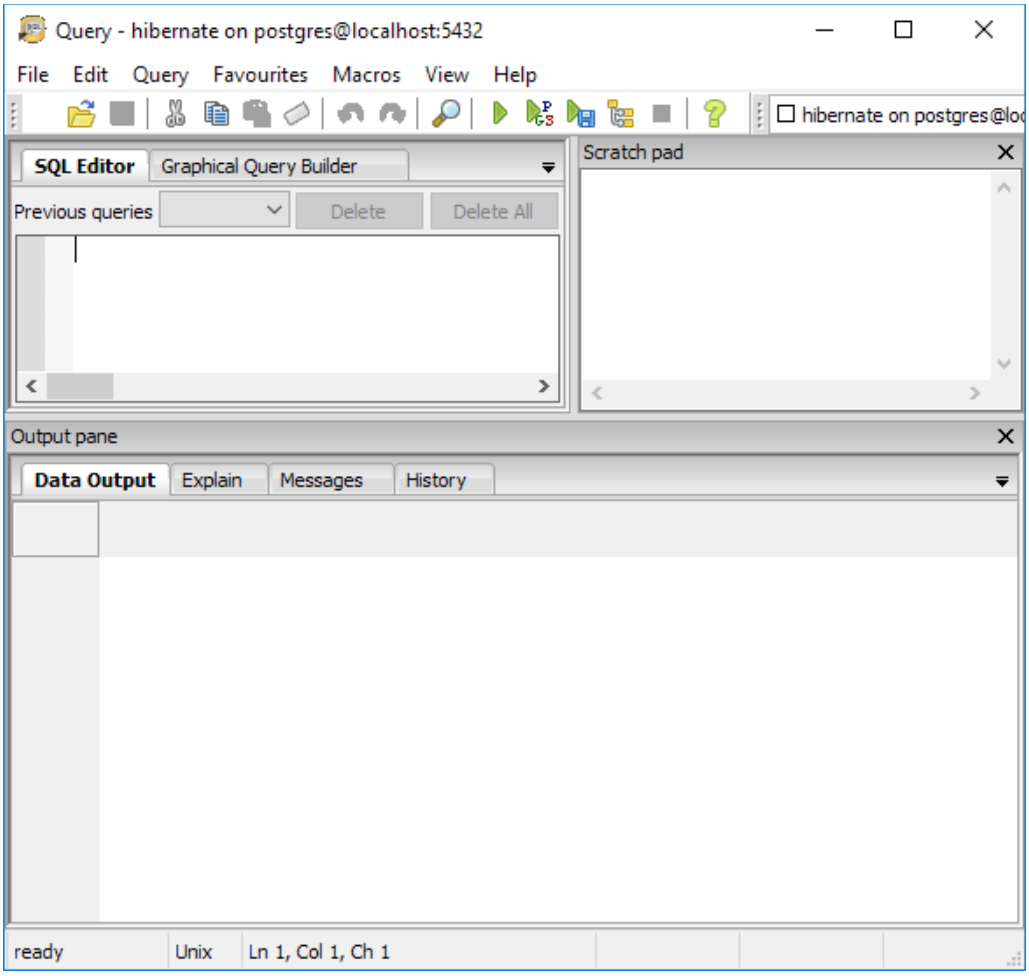
1. Double-click on the following:

`C:\Pentaho\postgresql\bin\pgAdmin3.exe`

2. Select and expand the hibernate database, then the Schemas.



3. Click on the SQL icon in the main toolbar.
4. The SQL editor will open.

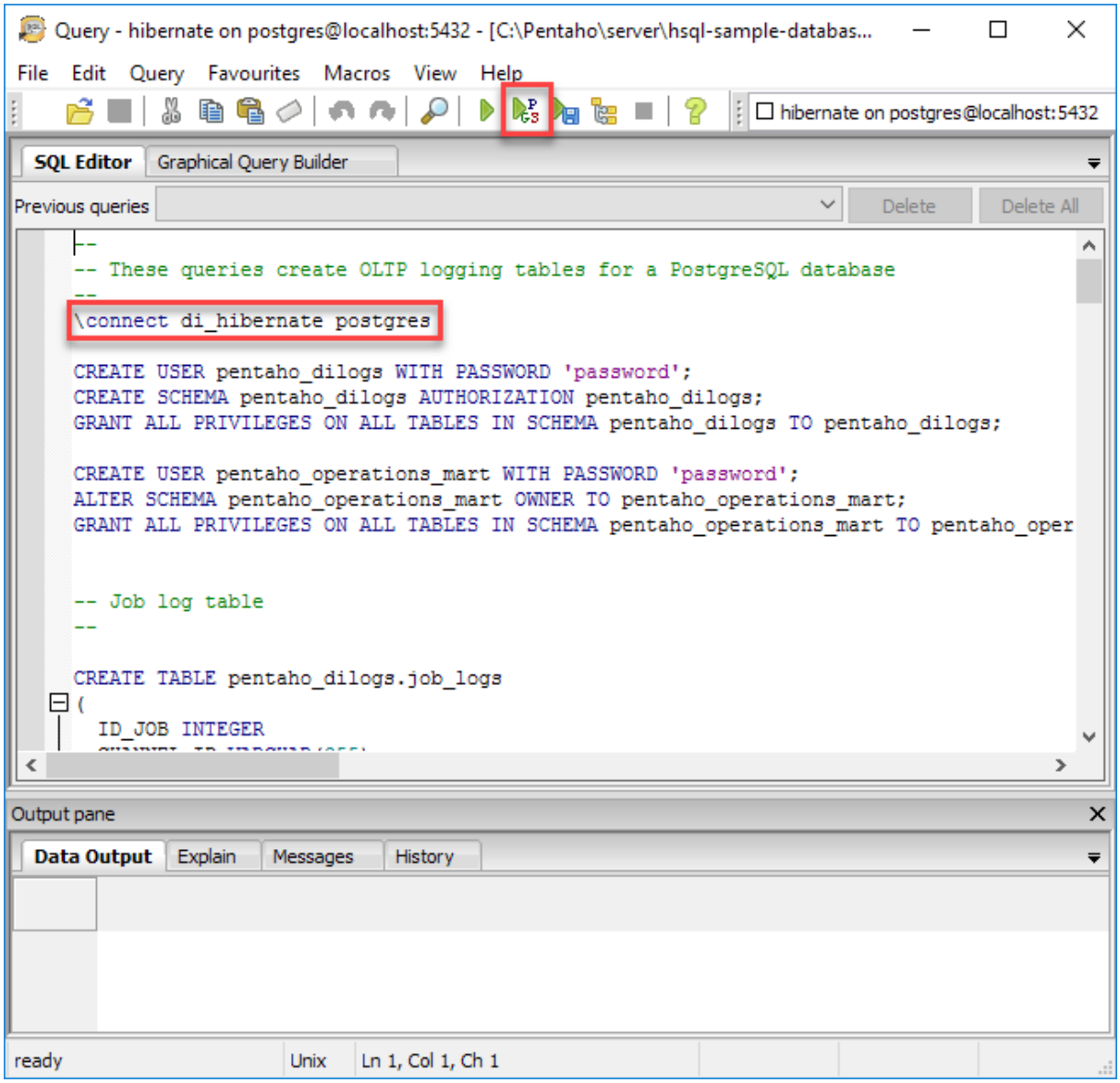


5. Browse for the following file.

   `C:\Pentaho\server\hsql-sample-database\postgresql\pentaho_logging_postgresql.sql`

6. Double click on the file to open in the SQL Editor
7. Remove the following line.

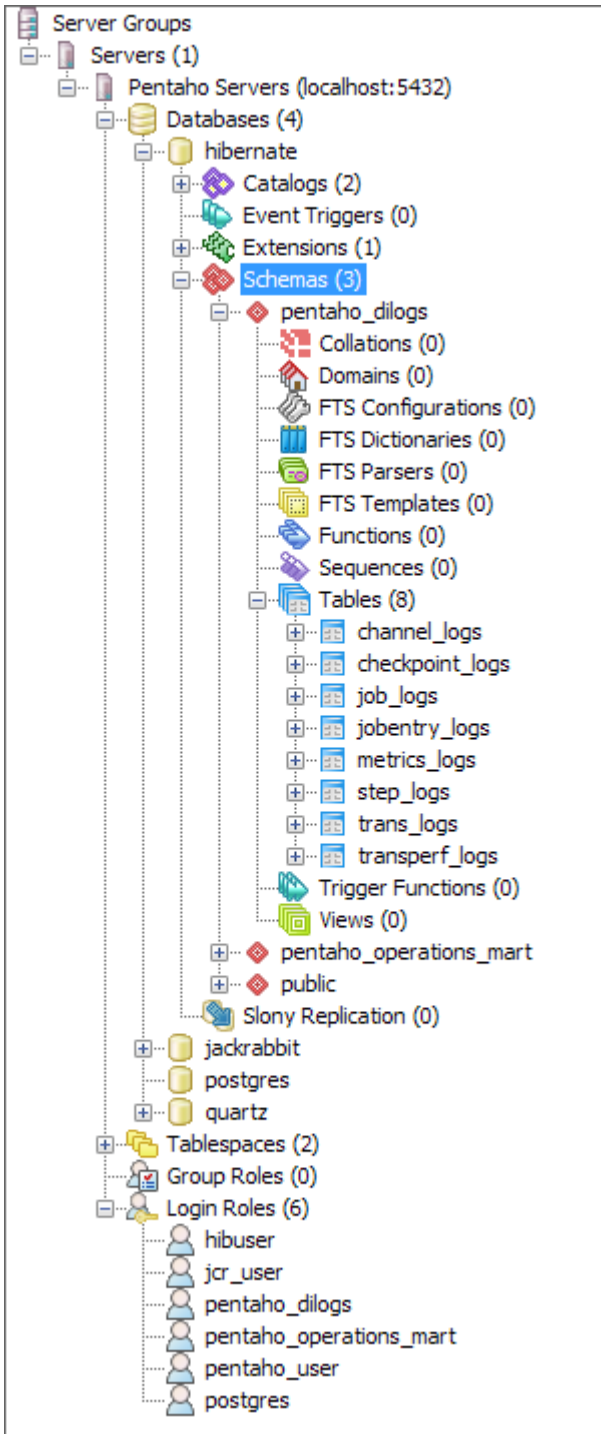
   `\connect di_hibernate postgres`



8. Click on the green PGS icon (as illustrated in the screenshot above) to execute the script.

With the release of Pentaho v7.1 the script has been updated. No modifications are required.

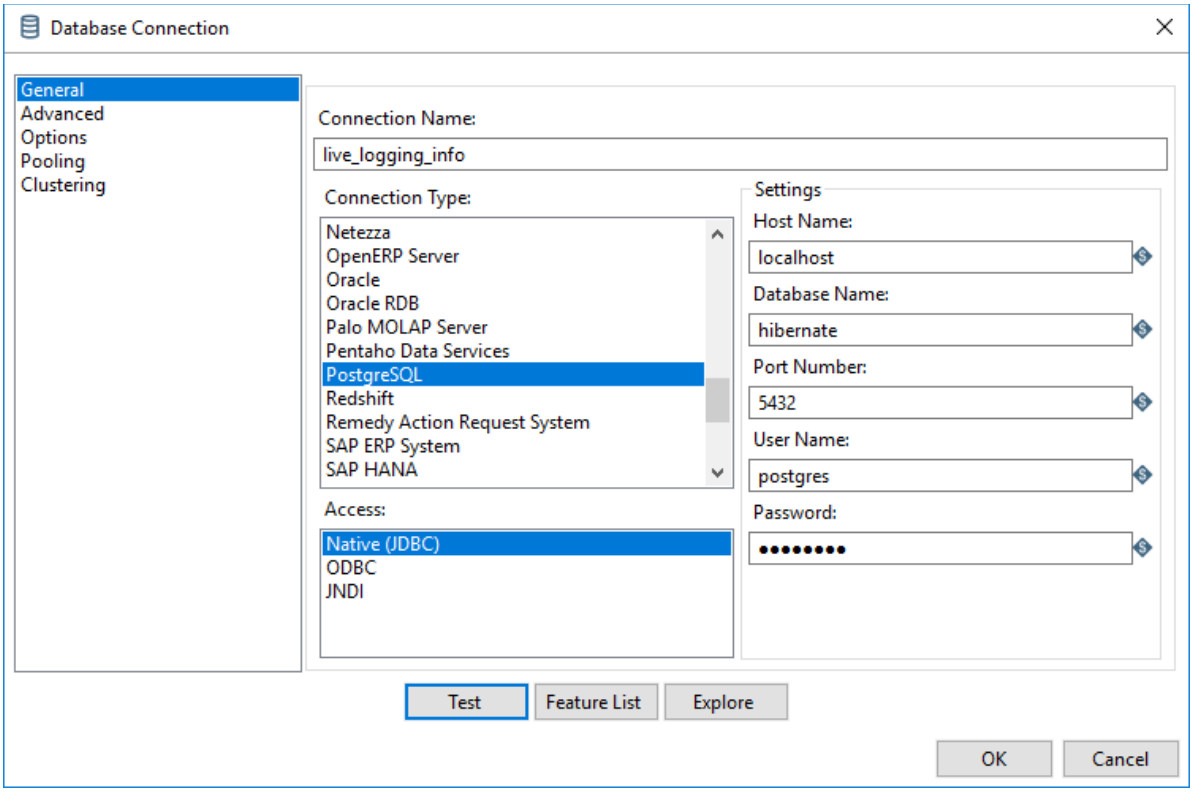
Check that the Schema, tables and Users have been correctly defined.



Next, in Spoon you will need to define a database connection to the HIBERNATE database.

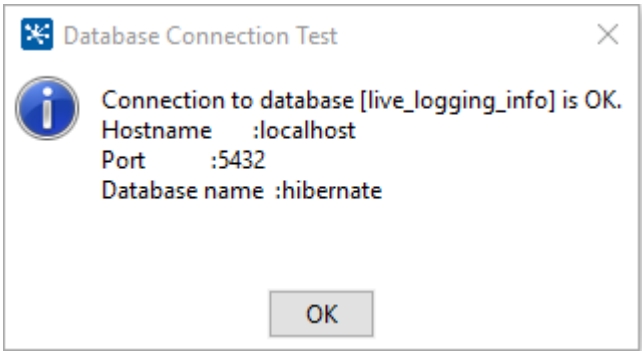
To define a database connection:

1. Click on the Create icon and from the drop-down list > Database Connection
2. Configure the connection as illustrated below:

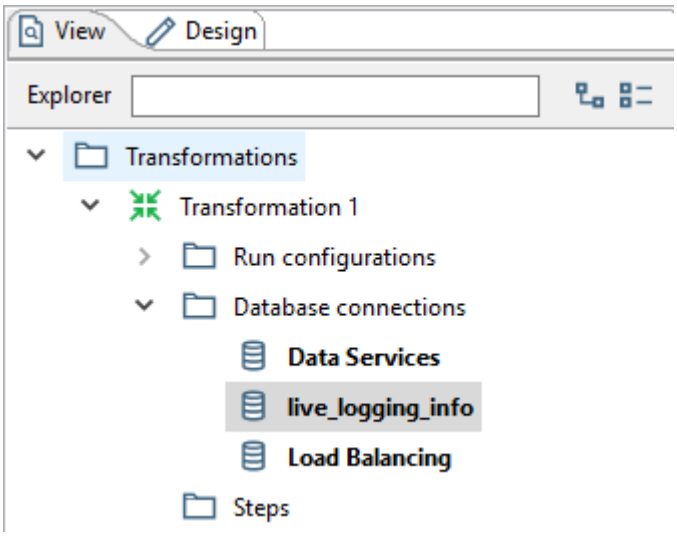


- The settings can be set as variables
- The kettle variables can be accessed by CTRL + SPACEBAR

3. Click on the Test button, if successfully connected, click OK to close.

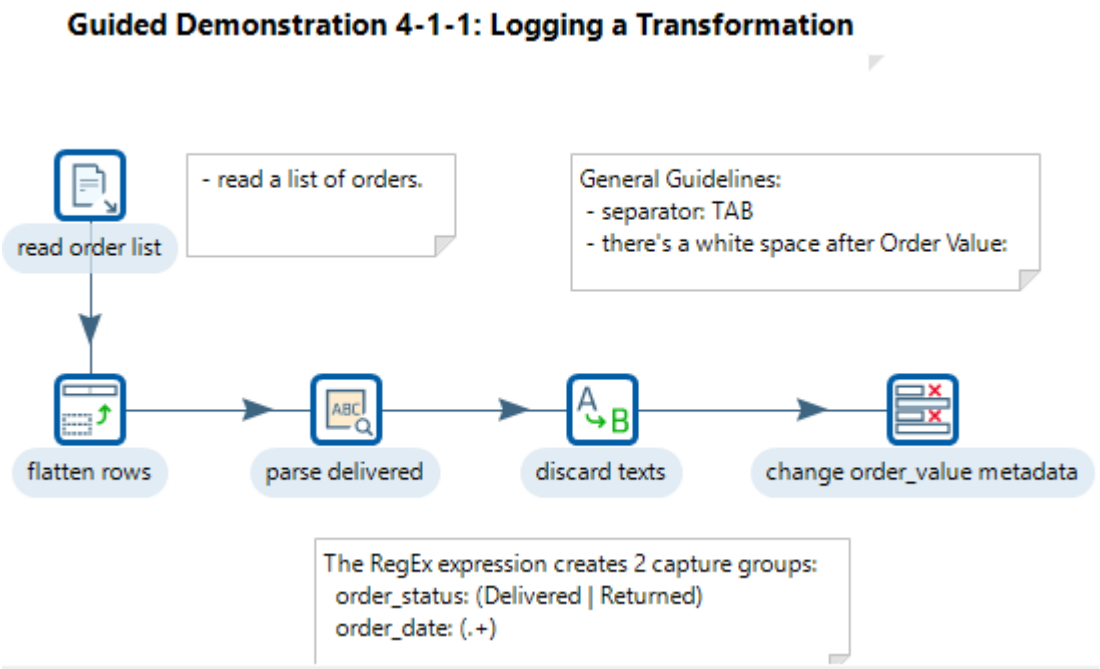


4. Ensure that the connection is shared:



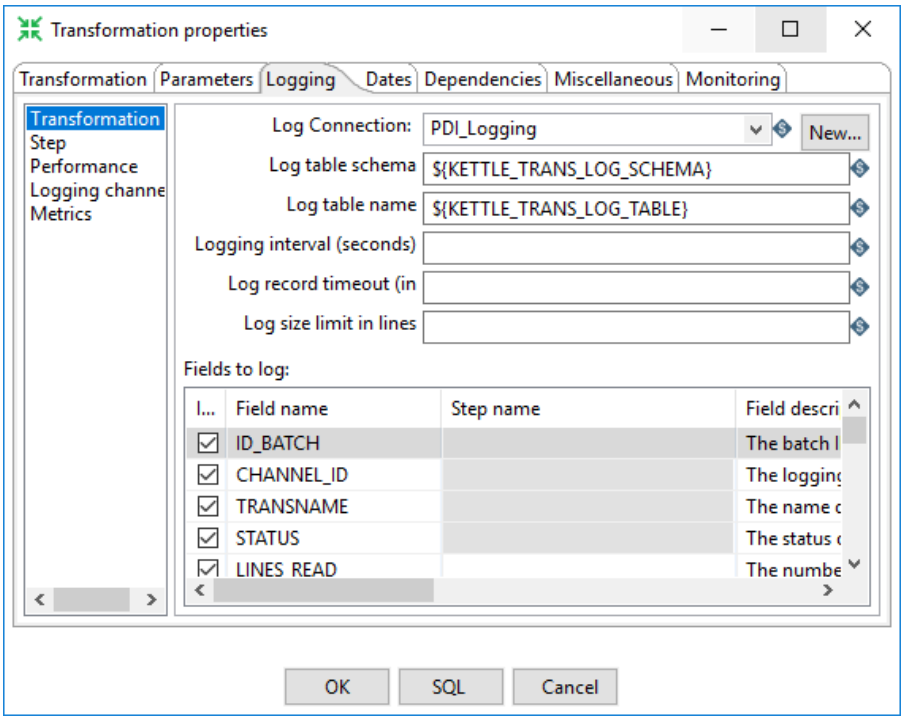
Let's configure a transformation for logging:

1. Open the logging transformation



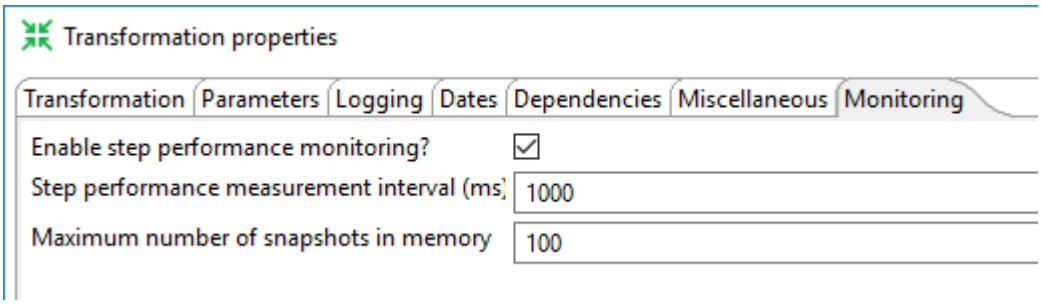
To configure the Transformation for logging:

2. Double click on the canvas to bring up the transformation properties
3. Click on the logging tab > Transformation. Ensure **all** the fields are selected.
4. Configure the Transformation option as illustrated below:



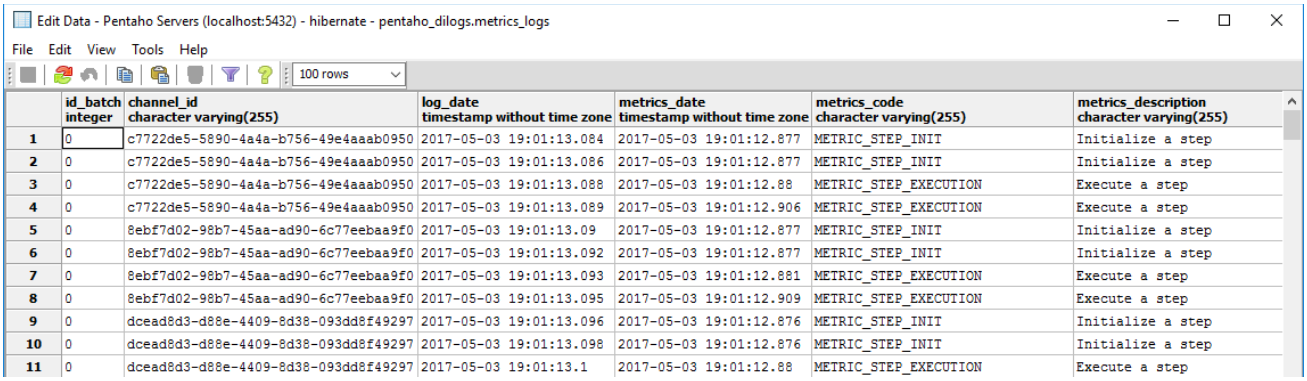
5. Repeat the workflow for the other options ensuring the correct variables are selected.

6. Click on the Monitoring tab and check the option: 'Enable step performance monitoring'



7. RUN the transformation several times.

Examine the Log tables

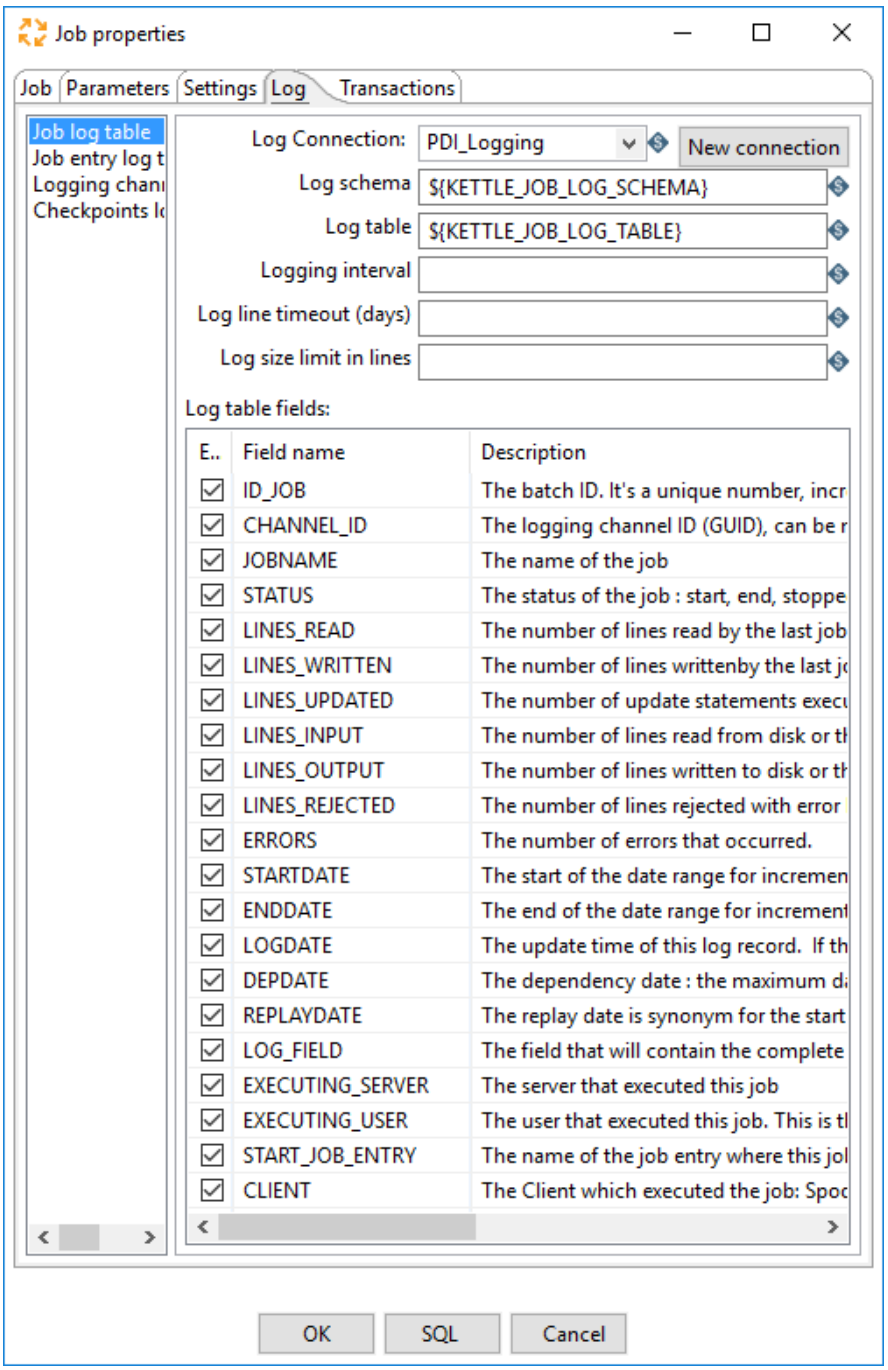


	id_batch integer	channel_id character varying(255)	log_date timestamp without time zone	metrics_date timestamp without time zone	metrics_code character varying(255)	metrics_description character varying(255)
1	0	c7722de5-5890-4a4a-b756-49e4aaab0950	2017-05-03 19:01:13.084	2017-05-03 19:01:12.877	METRIC_STEP_INIT	Initialize a step
2	0	c7722de5-5890-4a4a-b756-49e4aaab0950	2017-05-03 19:01:13.086	2017-05-03 19:01:12.877	METRIC_STEP_INIT	Initialize a step
3	0	c7722de5-5890-4a4a-b756-49e4aaab0950	2017-05-03 19:01:13.088	2017-05-03 19:01:12.88	METRIC_STEP_EXECUTION	Execute a step
4	0	c7722de5-5890-4a4a-b756-49e4aaab0950	2017-05-03 19:01:13.089	2017-05-03 19:01:12.906	METRIC_STEP_EXECUTION	Execute a step
5	0	8ebf7d02-98b7-45aa-ad90-6c77eebaa9f0	2017-05-03 19:01:13.09	2017-05-03 19:01:12.877	METRIC_STEP_INIT	Initialize a step
6	0	8ebf7d02-98b7-45aa-ad90-6c77eebaa9f0	2017-05-03 19:01:13.092	2017-05-03 19:01:12.877	METRIC_STEP_INIT	Initialize a step
7	0	8ebf7d02-98b7-45aa-ad90-6c77eebaa9f0	2017-05-03 19:01:13.093	2017-05-03 19:01:12.881	METRIC_STEP_EXECUTION	Execute a step
8	0	8ebf7d02-98b7-45aa-ad90-6c77eebaa9f0	2017-05-03 19:01:13.095	2017-05-03 19:01:12.909	METRIC_STEP_EXECUTION	Execute a step
9	0	dcead8d3-d88e-4409-8d38-093dd8f49297	2017-05-03 19:01:13.096	2017-05-03 19:01:12.876	METRIC_STEP_INIT	Initialize a step
10	0	dcead8d3-d88e-4409-8d38-093dd8f49297	2017-05-03 19:01:13.098	2017-05-03 19:01:12.876	METRIC_STEP_INIT	Initialize a step
11	0	dcead8d3-d88e-4409-8d38-093dd8f49297	2017-05-03 19:01:13.1	2017-05-03 19:01:12.88	METRIC_STEP_EXECUTION	Execute a step

A description for each of the Transformation and Job logging fields can be found under each corresponding Log tab.

To configure for logging a job:

1. Open the logging Job – GD3-4-1_logging_job.kjb
2. Double click on the canvas to bring up the job properties
3. Click on the Log tab
4. Highlight the Log Job table. Ensure all the fields are selected.
5. Configure the Job option as illustrated below:



6. Repeat the workflow for the other options ensuring the correct variables are selected.