

Basic Guide for the Installation and Use of the Extraction, Transformation and Loading Process Tool (ETLs)

Pentaho Data Integration (PDI) 10

Authors: Alejandro Maté Morgia
Juan Carlos Trujillo Mondéjar

Fecha: January 2025



Universitat d'Alacant
Universidad de Alicante



Máster de Formación Permanente
en **BIG DATA**
e Inteligencia Artificial



1. Introduction

Pentaho Data Integration Community Edition is an open source tool oriented towards the development and execution of Extraction, Transformation, and Load processes (ETL). PDI is one of the strong selling points of the Pentaho Analytics Suite. Most professionals recognize it as one of the better tools for data preparation, manipulation and load available in the market, surpassing paid tools offered by other competitors.

Pentaho Data Integration is a data flow creator. Data within the data flows is read from a varying number of sources, connected through one or more transformation steps, and loaded into one or more destination sinks, such as tables, files, document databases or other data persistence solutions. All data that passes through data flows is structured into registers, which are processed on a first come first processed basis, allowing the interested designer to optimize the performance of the ETL processes.

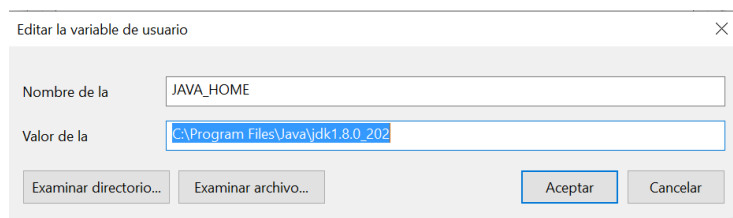
Although data is processed as structured registers, it is important that we do not assume that PDI is a tool that can only process traditional, structured data. It includes a wide array of connectors, source steps, transformations and sinks, which enable the ETL designer to face both traditional business data as well as Big Data sources. Moreover, in addition to the strong connectivity that PDI has, since its most recent versions it includes the possibility to deploy the processes as jobs over Apache Spark clusters. Nevertheless, deploying ETLs as Apache Spark jobs is an advanced topic that requires in-depth knowledge about both PDI as well as Spark and, thus, is considered to be outside of this guide.

2. Download and installation of PDI

2.1.- JAVA Pre-requisites.

To begin with, it is necessary that **Java 23** is installed and configured in the system. Superior versions can be used, but compatibility issues may arise. It is possible that it may work with older versions.

It is important to note that, similarly to how other applications that use JAVA, Pentaho Data Integration requires that the **JAVA_HOME** environment variable is defined. This variable must point to the path where Java is installed. By default, Pentaho runs a script that looks for the path and defines the **JAVA_HOME** variable. However, in some cases the script may not find the folder, leading to problems to start the application. In these cases, the best solution is to manually define the environment variable or to use a script such as: "for /d %%i in ("%Program Files\Java\jdk*") do set JAVA_HOME=%%i", assigning the variable **JAVA_HOME** to the most recent java version.



Another way to define the JAVA_HOME environment variable is through the environment variable editor and add the JAVA_HOME variable pointing to the directory where the jdk was unpacked.

2.2.- Installing PDI (Pentaho Data Integration)

Once Java is installed, PDI 10.X Community Edition can be downloaded from the following address:

<https://pentaho.com/pentaho-developer-edition/>

From this website, you can access all of the company's products, including the Pentaho package. The most downloaded products to design a Data Warehouse/Data Mart are: (i) Pentaho Schema Workbench, (ii) Pentaho Data Integraton (PDI), and (iii) Pentaho Server. In this guide, we will focus on the PDI, since it is one of the most used tools for data transformation due to its simplicity and friendly graphical environment.

Note: In order to facilitate the download, please go straight to the external link on the main tag of the course content.

Nombre	Fecha de modificación	Tipo	Tamaño
classes	27/12/2022 15:15	Carpeta de archivos	
Data Integration.app	27/12/2022 15:15	Carpeta de archivos	
Data Service JDBC Driver	27/12/2022 15:15	Carpeta de archivos	
docs	27/12/2022 15:15	Carpeta de archivos	
launcher	27/12/2022 15:15	Carpeta de archivos	
lib	27/12/2022 15:15	Carpeta de archivos	
libswt	27/12/2022 15:15	Carpeta de archivos	
plugins	27/12/2022 15:15	Carpeta de archivos	
pwd	27/12/2022 15:15	Carpeta de archivos	
samples	27/12/2022 15:15	Carpeta de archivos	
simple-jndi	27/12/2022 15:15	Carpeta de archivos	
static	27/12/2022 15:15	Carpeta de archivos	
ui	27/12/2022 15:15	Carpeta de archivos	
Carte	27/12/2022 15:14	Archivo por lotes de ...	2 KB
carte.sh	27/12/2022 15:14	Archivo SH	2 KB
Encr	27/12/2022 15:14	Archivo por lotes de ...	2 KB
encr.sh	27/12/2022 15:14	Archivo SH	2 KB
Import	27/12/2022 15:14	Archivo por lotes de ...	2 KB
import.sh	27/12/2022 15:14	Archivo SH	2 KB
import-rules	27/12/2022 15:14	Documento XML	3 KB
Kitchen	27/12/2022 15:14	Archivo por lotes de ...	2 KB
kitchen.sh	27/12/2022 15:14	Archivo SH	2 KB
LICENSE	27/12/2022 15:14	Documento de texto	14 KB
Pan	27/12/2022 15:14	Archivo por lotes de ...	2 KB
pan.sh	27/12/2022 15:14	Archivo SH	2 KB
PentahoDataIntegration_OSS_Licenses	27/12/2022 15:15	Firefox HTML Docum...	3 KB
purge-utility	27/12/2022 15:14	Archivo por lotes de ...	2 KB
purge-utility.sh	27/12/2022 15:14	Archivo SH	2 KB
README	27/12/2022 15:14	Documento de texto	2 KB
runSamples	27/12/2022 15:14	Archivo por lotes de ...	2 KB
runSamples.sh	27/12/2022 15:14	Archivo SH	2 KB
set-pentaho-env	27/12/2022 15:14	Archivo por lotes de ...	6 KB
set-pentaho-env.sh	27/12/2022 15:14	Archivo SH	5 KB
Spoon	27/12/2022 15:14	Archivo por lotes de ...	6 KB
spoon	27/12/2022 15:14	Archivo COMMAND	2 KB
spoon	27/12/2022 15:14	Icono	204 KB
spoon	27/12/2022 15:14	Archivo PNG	1 KB
spoon.sh	27/12/2022 15:14	Archivo SH	9 KB
SpoonConsole	27/12/2022 15:14	Archivo por lotes de ...	2 KB
SpoonDebug	27/12/2022 15:14	Archivo por lotes de ...	3 KB
SpoonDebug.sh	27/12/2022 15:14	Archivo SH	2 KB
yarn.sh	27/12/2022 15:14	Archivo SH	2 KB

Figure 1. Pentaho Data Integration Folder

In this folder we can find all the files of Pentaho Data Integration, including multiple sample flows that provide an example of how the different transformation steps work, as we will show afterwards.

Aside from these requirements, the most relevant files found in the PDI folder (.bat for Windows and .sh for Linux) are as follows:

- **Spoon:** It is the main file that runs the Graphic User Interface (GUI) of PDI. This file is the one that we will **most commonly** use, since the GUI facilitates the process of creating and running data flows and PDI is not designed to be interacted and run from a programming interface.
- **Pan:** Pan is a file designed to run transformations using the command line. Pan takes as input a PDI transformation created with Spoon, and runs it through the command line, using the terminal as output log until the transformation finishes. When running under Linux, we can redirect the output log to a file by specifying the ">" redirection command, and execute run the transformation in a non-blocking fashion with the "&" command.



- **Kitchen:** Kitchen is similar to Pan, but in this case it is a program designed to run a job instead of a transformation. The rest of its functionality is identical to Pan.
- **Carte:** Carte is a transformation and job server, designed to run jobs and transformation as a service. The main goal of Carte is to enable the execution of transformations and jobs in remote computers. This allows us to use our computer as design center for transformations, running them in a more powerful or dedicated server instead. In order to enable the traceability of transformations and jobs Carte also serves a webpage that can be accessed by the user and shows all the information about the status and events related to the transformation and jobs run in the server.

It is likely that, at this point, the reader is asking herself the question Why would I need a command line software in a GUI-oriented ETL tool? The answer is that, even if the interface that PDI offers is simple and user-friendly, the stability of the GUI when under heavy data loads suffers significantly. It is sufficient running any slightly complex transformation that processes above millions of registers and it is likely that the software will crash, making it impossible to successfully finish the transformation. If the magnitude of data that we need to process reaches thousands of millions of registers (likely in Big Data environments), then no matter how much memory we assign to PDI, it will be necessary to run our transformation using Kitchen or Pan.

2.3.- Copying the right database drivers in the \lib folder

Before we continue, we should remind that we need the right database driver in the lib folder. Actually, you can see that PDI comes with many drivers. For example, if we are connecting an Oracle database, we need to download the Java Oracle database driver and copy it into the ...`\data-integration\lib\` folder. You can download any Oracle connector from the following URL:

<https://www.oracle.com/es/database/technologies/appdev/jdbc-downloads.html>

As we previously mentioned, we recommend downloading the `ojdbc8.jar` as the `ojdbc11.jar` may cause some problems with some Operating systems and PDI installations.

3. Development environment

Some of the basics to start using Pentaho Data Integration will be explained below. In order to access the graphic design interface we must run **Spoon.bat**, as indicated above. When opening Spoon, after a few moments, we will see an interface like the one shown in Figure 3.

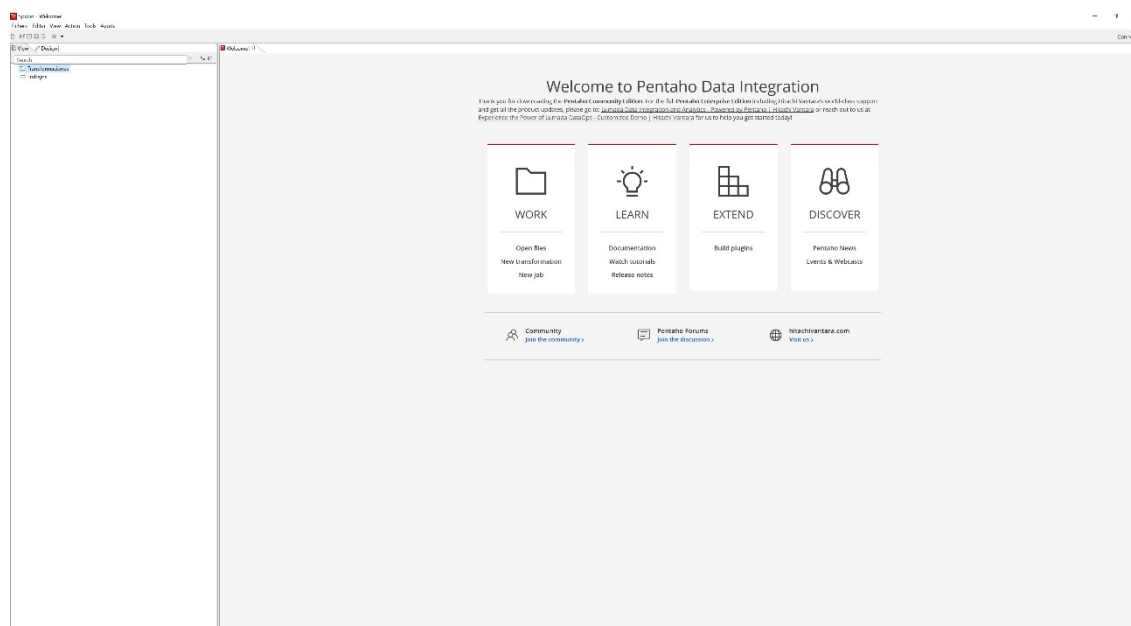
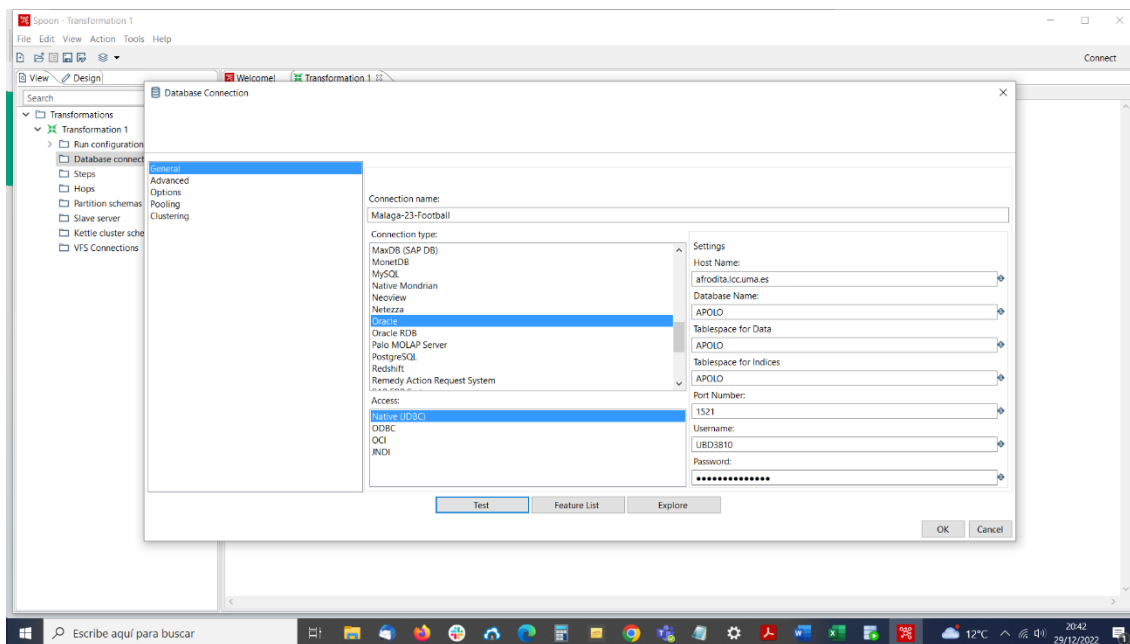


Figure 2. Pentaho Data Integration landing page

3.1.- Checking the database connection

Once the right database drives have already been copied into the ...\\lib folder, we may wish to create a new Database Connection to our database. We should remember that even though the database connection is failed, we can still continue transforming data from and to local files.

In the following figure, we can see the parameters we need to introduce for the database connection to our database:



To start working, just select the File-> New-> Transformation option, and we can start designing our transformation.

When we create a new transformation, a screen like the one shown in Figure 4 will appear, in which we are instructed to drag elements from the step palette into the design area.

The steps are organized in thematic folders in PDI, the most commonly used are those of Input (Input), Output (Output), Union (Joins), and Transformation. The power of information extraction becomes evident in PDI simply by opening the Input folder. In Figure 5 we can see that, on the basis, PDI supports dozens of possible sources of data, including the generation of random values in case we need to simulate data flows of different or even infinite volumes.

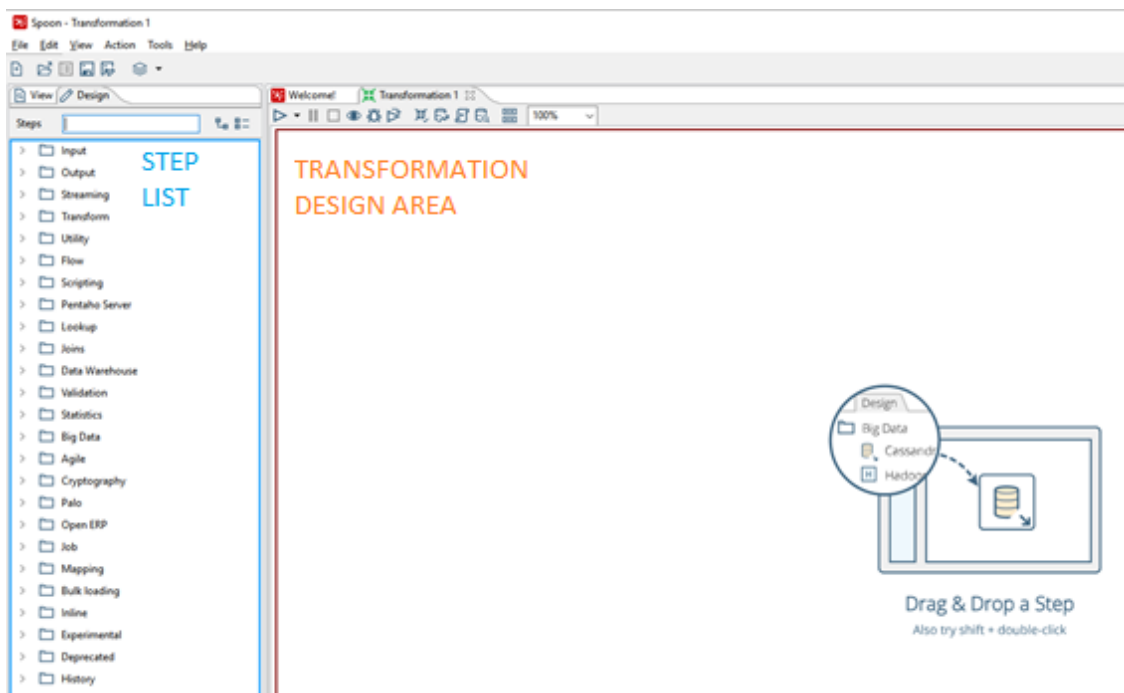


Figure 3. Development environment for transformations

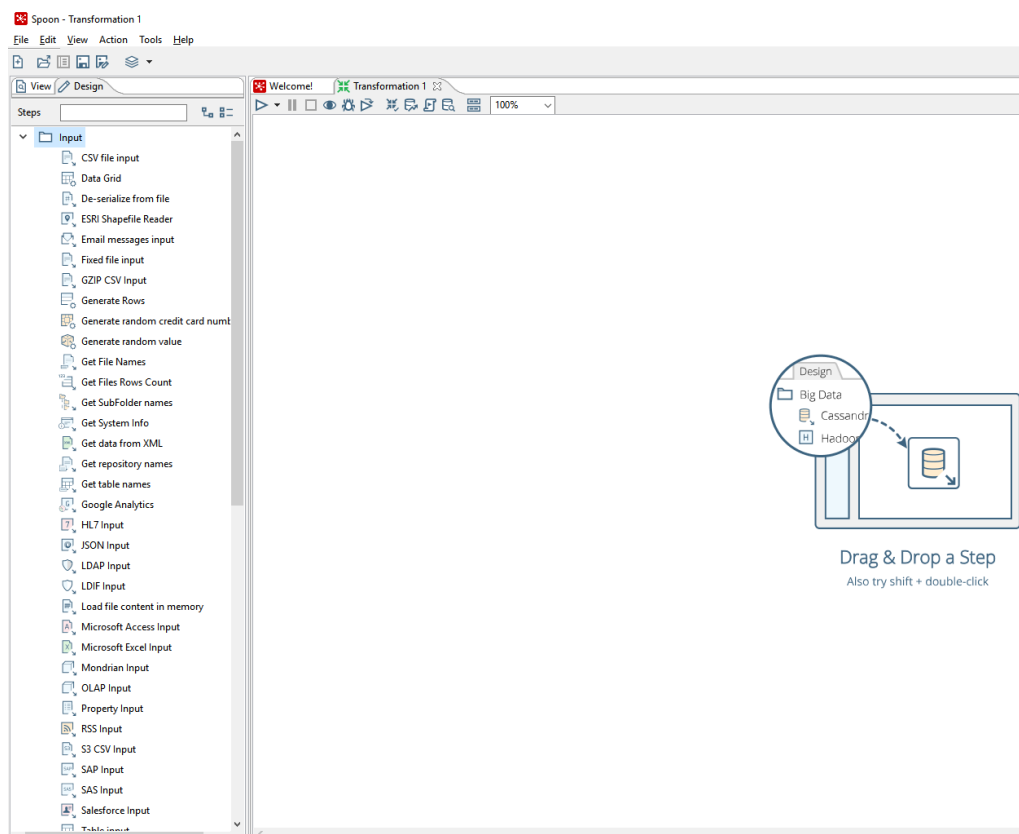


Figure 4. Input steps available in PDI

To exemplify a typical data manipulation process, this introduction guide will use the call file to the San Francisco fire department. This data set is a relatively high volume set (1.5 GB and more than 4 million entries).

The process that we are going to carry out will be the following: We will extract from the data set, without a priori information from other sources, what are the different neighborhoods that are in the file and the number of calls that each one contains. Once this data is obtained, we will save it as output CSV files.

To start, let's read the call file. The file is a CSV, so we will use the **CSV Input step**. First, we drag the CSV Input step from the step palette to the design area. Next, we double click on it to open the configuration window. In Figure 6 we can see the configuration window of the CSV step.

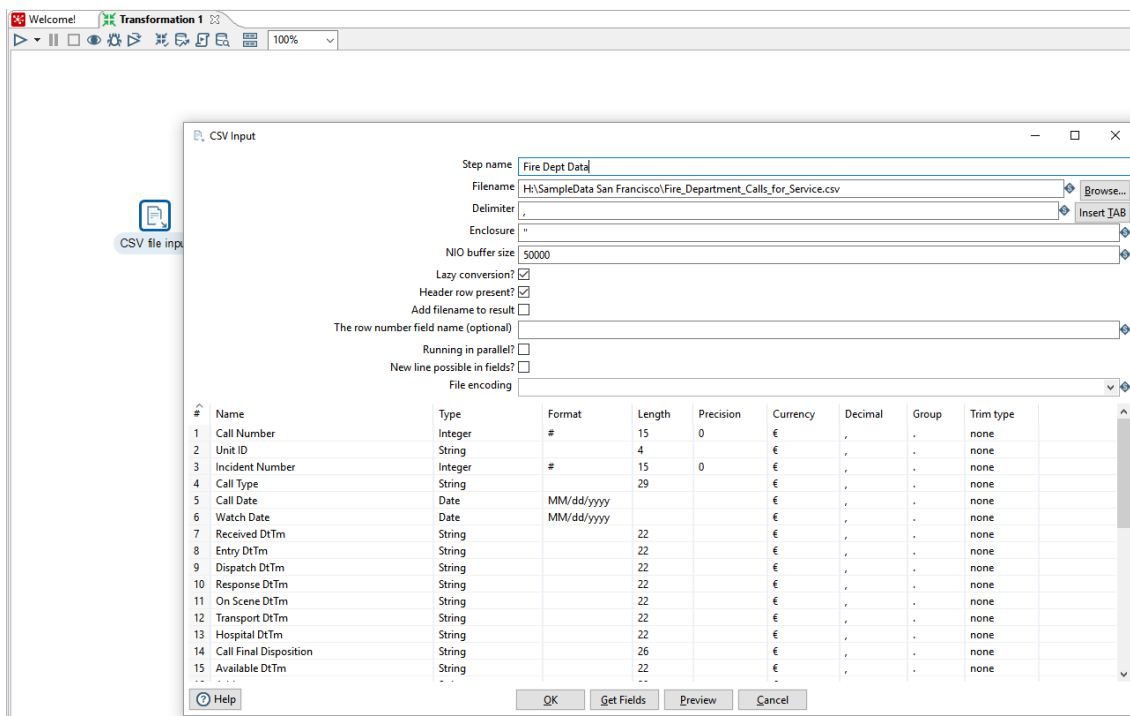


Figure 5. CSV Input step

In order to read the data from the file, we must first select the file from which we want to read. The file selection is made using the Browse ... button, which is located to the right of the file name.

Once the file is selected, we must analyze what its **content** is to identify the columns and their data types. PDI performs this task for us by limited sampling of the data. To do this, we click on the **Get Fields button**, located at the bottom of the window, and indicate the number of rows to analyze. The more rows we analyze, the more accurate the results and the worse the execution time. If we are satisfied with the results, press OK and we will have the data entry step configured.

Since we only want to analyze in this example the data referring to the districts and the number of calls, we will reduce our data set as soon as possible. To do this, we will use the **Select values step**, which allows us to alter the structure of our data **to keep only part of it or alter its type of data**. As in the previous case, we drag the step to the design area. Once the step has been added, we must connect it with the previous step so that the metadata (the information regarding which columns we have to be able to operate) can be read by our Select values step.

To connect two steps, simply keep the **Shift keypad button while selecting and dragging from the source step to the destination step**. The result should be similar to Figure 7.

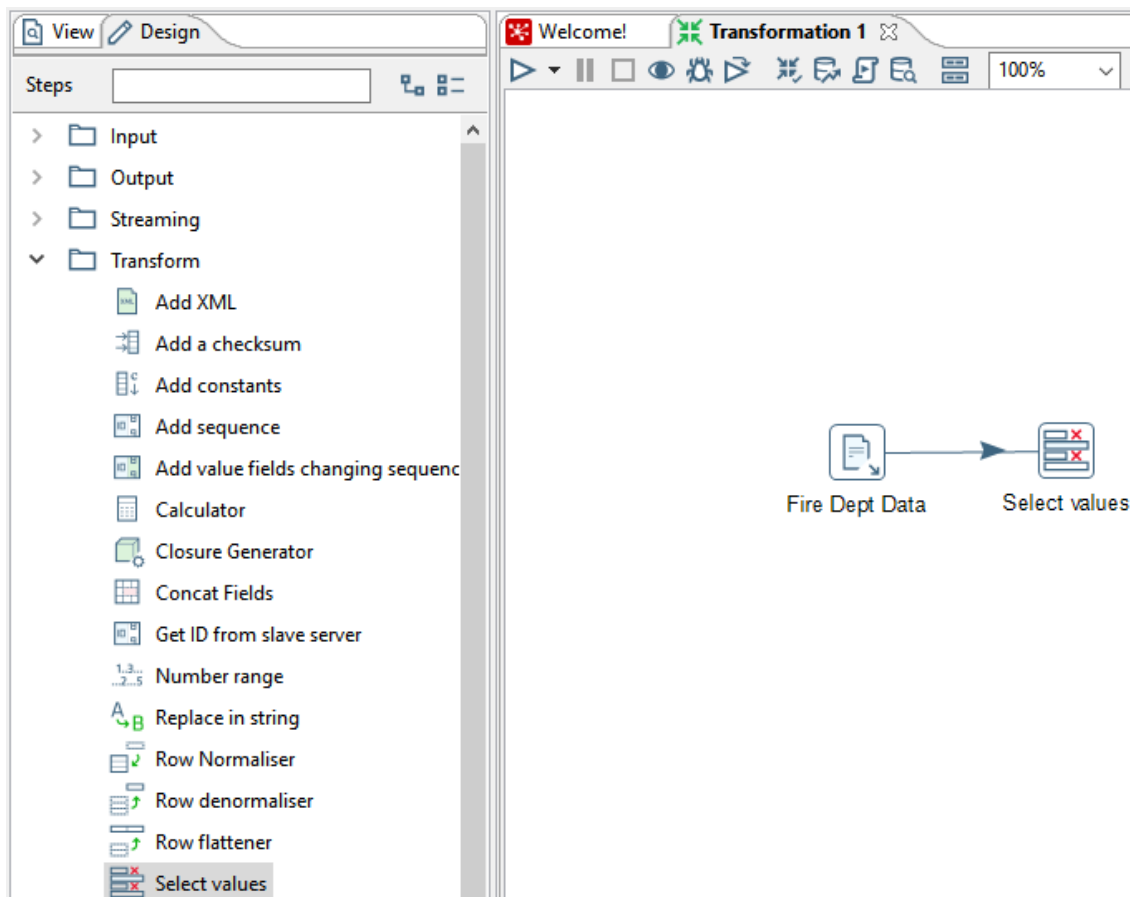


Figure 6. Connection between the CSV Input Step and Select Values

Once the steps are connected, we can configure our Select values step. As in the previous case, just double click on it and use the **Get fields** to select button, located on the right side of the dialog that appears.

The Select values step allows you to operate in different ways with the available fields. In our case, the easiest way will be to **select only the data that we want**, in this case those corresponding to the neighborhood, as shown in Figure 8. Selecting the neighborhoods will not delete the repeated records, it will cause that, from this the moment the flow only contains the columns corresponding to the neighborhoods, thus having more than 4 million records with repeated neighborhoods, one respecting one record per call.

Once we have the list of neighborhood records, we will **calculate how many calls there are per neighborhood**. To do this, we will first use **the Sort rows step** and then the **Group by** step.

We will repeat the same operation as before, first adding the **Sort rows step** and **connecting it with our Select values step**. The configuration of the Sort rows step is shown in Figure 9.

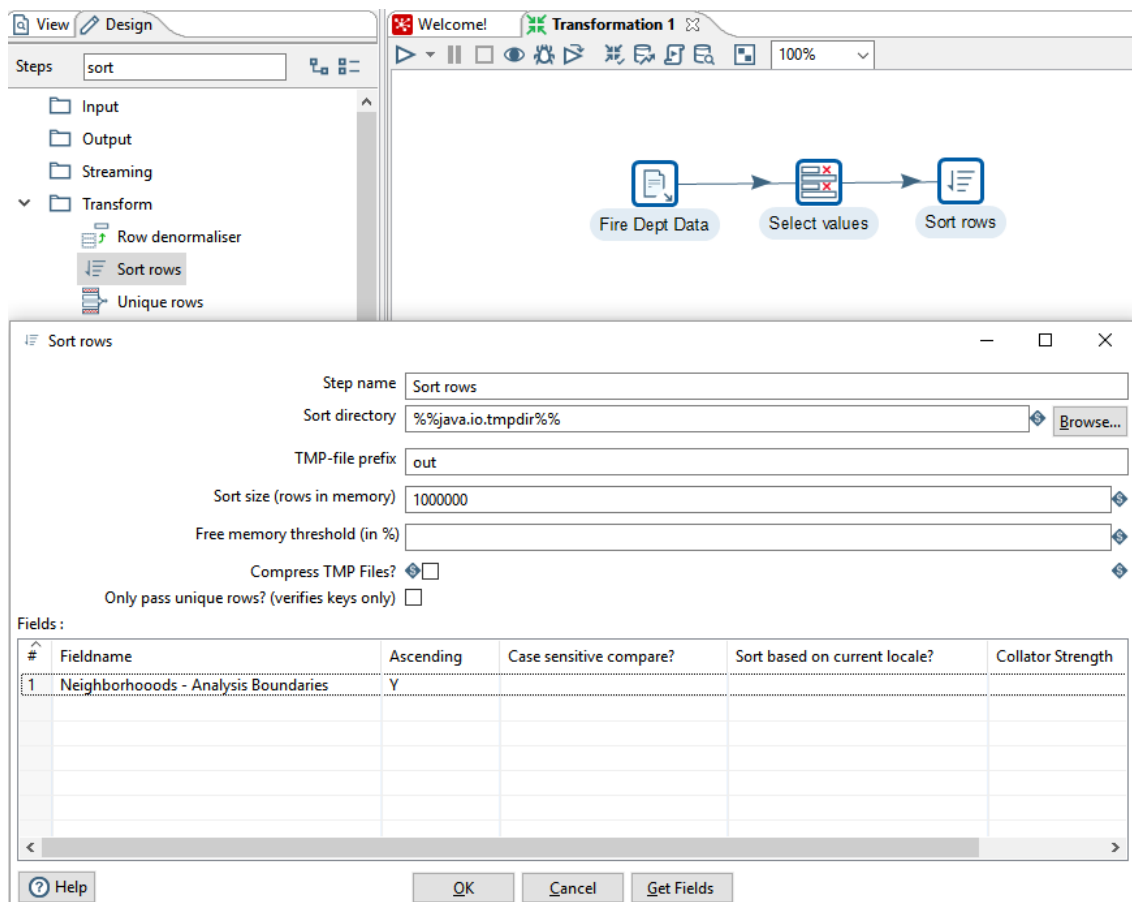


Figure 8. Sorting neighborhood rows

Once we have the data ordered, we only need to **calculate the number of calls per neighborhood**. This step will result in both a list of **unique** neighborhood values along with your total calls for each. To do this, we add a **Group by step** and connect it to the **Sort rows** step.

The configuration of the **Group by** step can be a bit confusing for new developers. **At the top** you must place the fields that correspond to the aggregation you want to perform. In our case here we will place the neighborhoods. Any input field not included in this section will be removed from the output.

On the other hand, **the calculations to be performed must be indicated in the lower area**. Here the **first column** represents the new fields to be created, **the second column** the field on which to perform operations (for sums, means, etc.), and the **third column** the type of operation to be performed. For our calculations we will configure the Group by step as shown in Figure 10.

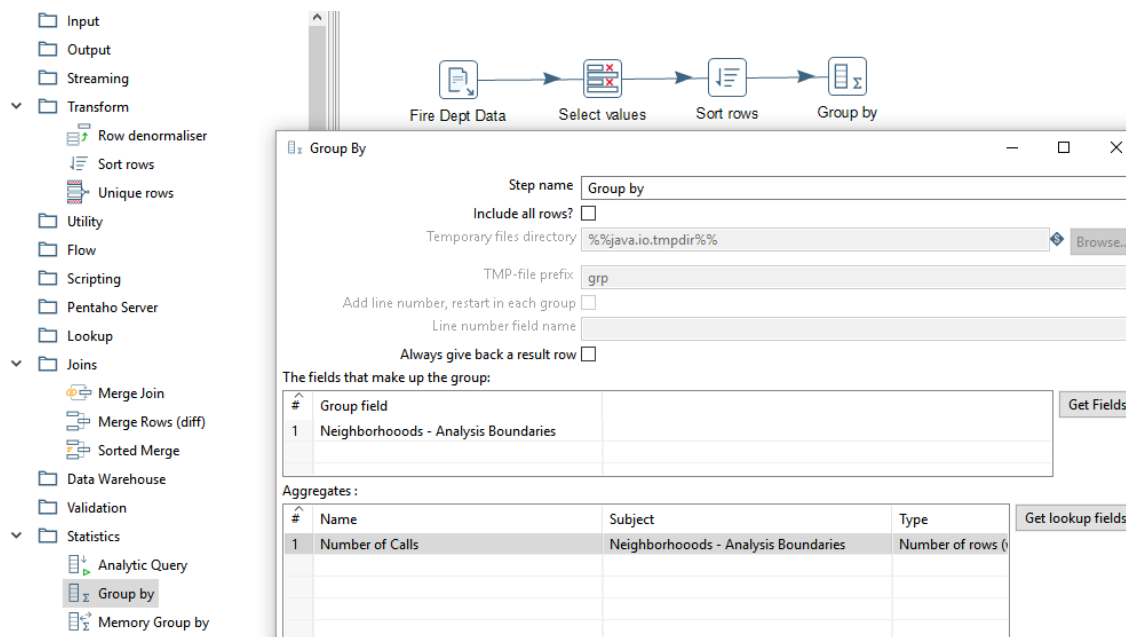


Figure 9. Calculating number of calls per neighborhood

To finish our transformation we will **save the results we have obtained in an output file, which we will call FireDeptResults.csv**. Output file management is carried out using the Text File Output step. **The Text File Output steps** allow you to save in text or CSV format and indicate different separator elements to separate the information from the different columns. The configuration of the Text File Output step can be seen in Figures 11 and 12. Similar to the Text File Input step, the Text File Output step requires that we retrieve the flow fields by pressing the **Get Fields** button on the Fields tab. Once this operation is done we can configure the metadata of the fields to be saved in case we want to modify them.

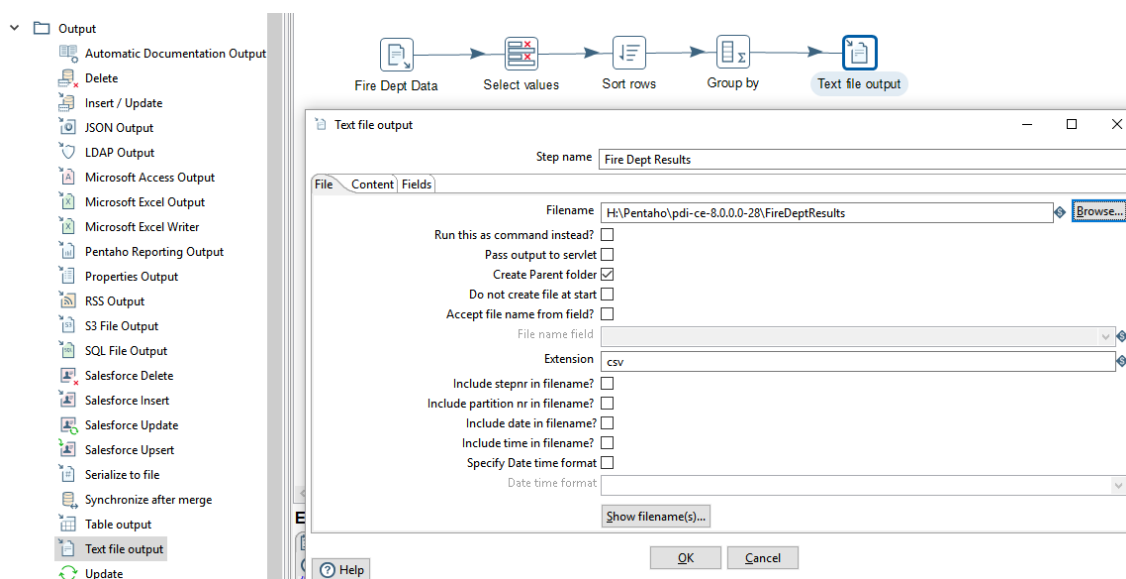


Figura 10. Saving results in a CSV file (1)

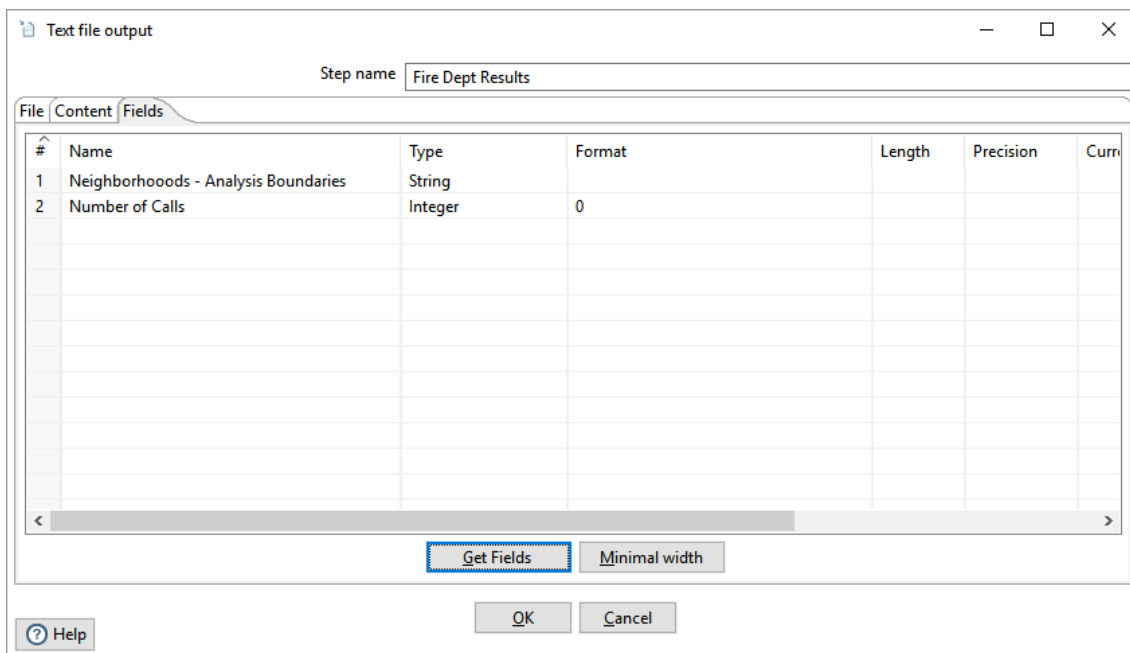


Figure 11. Saving results in a CSV file (2)

If we are satisfied with our transformation, we can proceed to test it using the “Play” arrow button at the top of the interface, as shown in Figure 13.

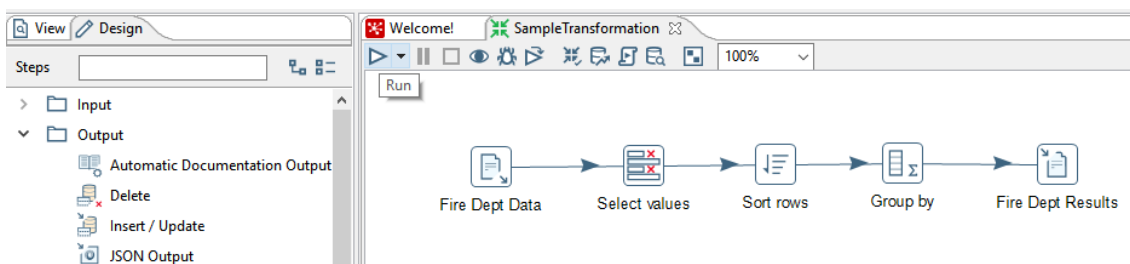


Figure 12. Transformation execution

When executing the transformation we can see the processing speed of the rows in each step, although we will have to take into account that a step can be slowed down by the steps after or before it since the data buffer is limited.

If everything went well we will have a result similar to that shown in Figure 14.

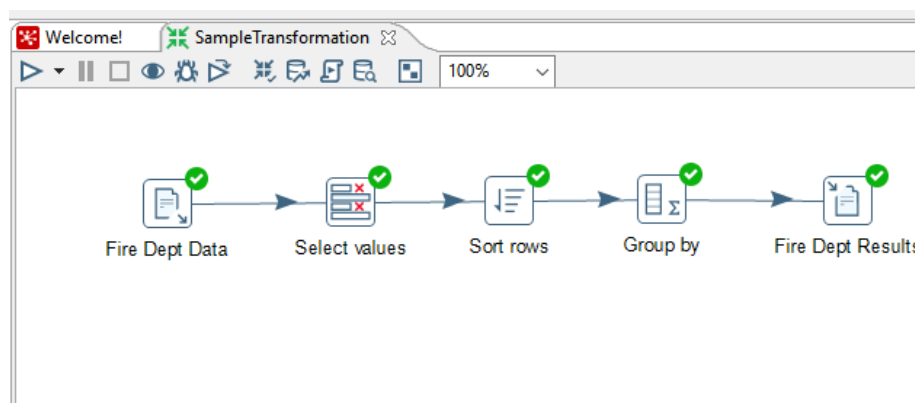
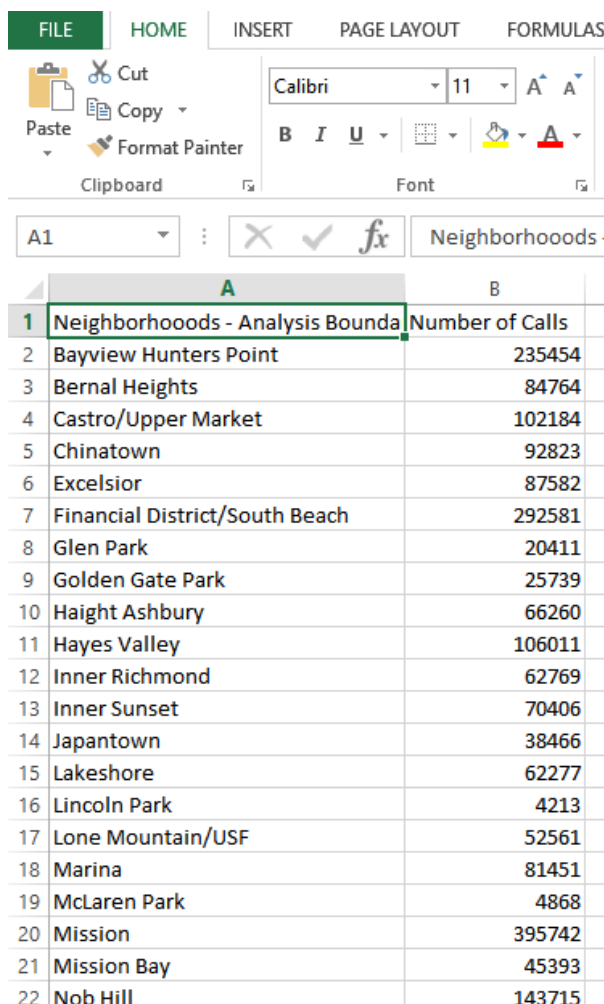


Figure 13. Successful execution of the transformation

When opening the CSV file with Excel or a similar program (Google Sheets, Open Office, etc.) we can see the results obtained, as shown in Figure 15. We have just processed more than 4 million rows automatically, being able to repeat the process with new data as many times as we want, as long as they follow the same format.



	A	B
1	Neighborhoods - Analysis Bounda	Number of Calls
2	Bayview Hunters Point	235454
3	Bernal Heights	84764
4	Castro/Upper Market	102184
5	Chinatown	92823
6	Excelsior	87582
7	Financial District/South Beach	292581
8	Glen Park	20411
9	Golden Gate Park	25739
10	Haight Ashbury	66260
11	Hayes Valley	106011
12	Inner Richmond	62769
13	Inner Sunset	70406
14	Japantown	38466
15	Lakeshore	62277
16	Lincoln Park	4213
17	Lone Mountain/USF	52561
18	Marina	81451
19	McLaren Park	4868
20	Mission	395742
21	Mission Bay	45393
22	Nob Hill	143715

Figure 14. Sample results obtained

With this transformation we conclude our example of introduction to Pentaho Data Integration. Next we will see some aspects that will allow to deepen the habitual use of PDI for the transformation of data in a general way.

3.1.- Transformations and Jobs

So far we have seen the Pentaho Data Integration **transformations**, which are composed of steps that allow us to carry out ETL processes. **The concept of Jobs in PDI results from hierarchically organizing the execution of ETL processes. The works are "transformations of transformations".** A job does not contain reading, data manipulation, or writing steps, but contains a starting point ("Start" step) and an ending step ("Success"). Among these steps will be, either calls to transformations and other works that we have previously designed, or auxiliary steps such as sending emails to

notify that there have been problems, manage directories and files, delete temporary, etc. We can see an example of a work in Figure 16.

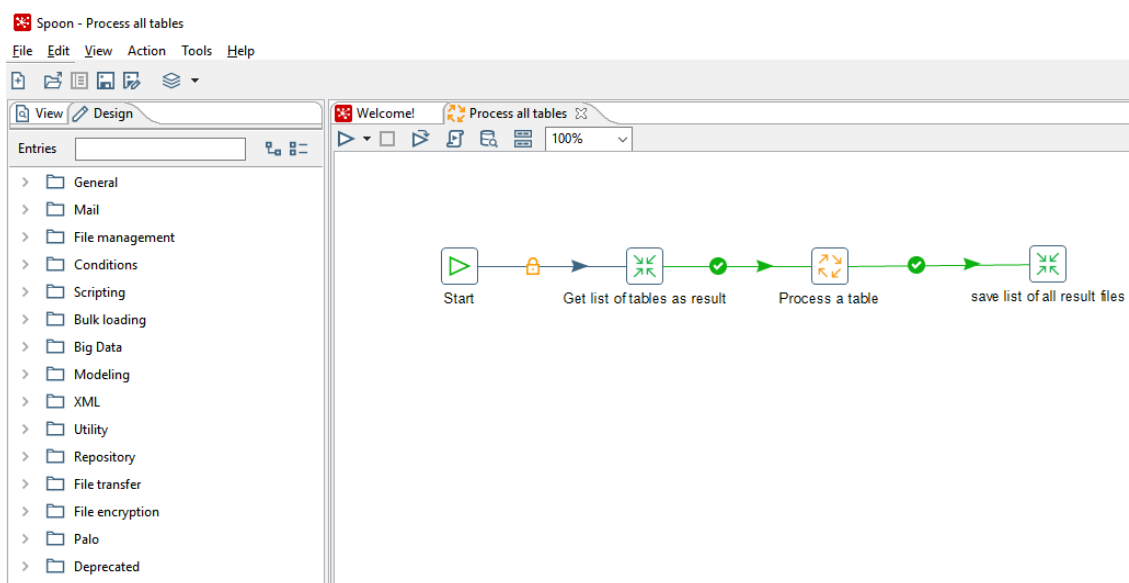


Figure 15. Example of a PDI Job

3.2.- Examples

Given the volume of steps included (not counting extensions) in Pentaho Data Integration, it is impossible to cover each guide in detail. To be able to find steps that are suitable for a variety of situations that a data engineer must face, you can either visit the Web <https://help.hitachivantara.com/Documentation/Pentaho/9.4> or open the transformations and jobs contained in the “samples” folder, within the data-integration directory:

Nombre	Fecha de modificación	Tipo	Tamaño
classes	27/12/2022 15:15	Carpeta de archivos	
Data Integration.app	27/12/2022 15:15	Carpeta de archivos	
Data Service JDBC Driver	27/12/2022 15:15	Carpeta de archivos	
docs	27/12/2022 15:15	Carpeta de archivos	
launcher	27/12/2022 15:15	Carpeta de archivos	
lib	27/12/2022 15:15	Carpeta de archivos	
libswt	27/12/2022 15:15	Carpeta de archivos	
logs	27/12/2022 15:49	Carpeta de archivos	
plugins	27/12/2022 15:15	Carpeta de archivos	
pwd	27/12/2022 15:15	Carpeta de archivos	
samples	27/12/2022 15:15	Carpeta de archivos	
simple	27/12/2022 15:15	Carpeta de archivos	
static	27/12/2022 15:15	Carpeta de archivos	
ui	27/12/2022 15:15	Carpeta de archivos	
Carte	27/12/2022 15:14	Archivo por lotes de ...	2 KB
carte.sh	27/12/2022 15:14	Archivo SH	2 KB
Encr	27/12/2022 15:14	Archivo por lotes de ...	2 KB
encr.sh	27/12/2022 15:14	Archivo SH	2 KB
Import	27/12/2022 15:14	Archivo por lotes de ...	2 KB
import.sh	27/12/2022 15:14	Archivo SH	2 KB
import-rules	27/12/2022 15:14	Documento XML	3 KB
Kitchen	27/12/2022 15:14	Archivo por lotes de ...	2 KB
kitchen.sh	27/12/2022 15:14	Archivo SH	2 KB
LICENSE	27/12/2022 15:14	Documento de texto	14 KB
Pan	27/12/2022 15:14	Archivo por lotes de ...	2 KB
pan.sh	27/12/2022 15:14	Archivo SH	2 KB
PentahoDataIntegration_OSS_licenses	27/12/2022 15:15	Firefox HTML Docum...	3 KB
purge-utility	27/12/2022 15:14	Archivo por lotes de ...	2 KB
purge-utility.sh	27/12/2022 15:14	Archivo SH	2 KB
README	27/12/2022 15:14	Documento de texto	2 KB
runSamples	27/12/2022 15:14	Archivo por lotes de ...	2 KB

Figure 16. Samples folder included in PDI

4. Further information

In this guide we have covered the fundamental aspects of Pentaho Data Integration. The design of the transformations and works occupies the vast majority of the work of a data engineer dedicated to ETL processes (in many cases 95% of the time). However, PDI includes other features that, although not detailed in this guide, may be of interest.

On the one hand, PDI has a Marketplace (Tools-> Marketplace), similar to Pentaho Server, where you can download plugins and additional steps that can simplify data processing.

On the other, PDI allows you to organize transformations and work in repositories, to facilitate your organization and avoid duplicates and errors in professional deployments. Finally, the connectivity features with distributed file systems (Hadoop FS) require for the connection the installation of Hadoop Shims, which abstract the problem of changing Hadoop versions and allow PDI to access data that is distributed in the system. distributed files.

It should be noted that there are additional features in the Enterprise version, such as queue processing or graphical analysis directly on data flows.

All these features are advanced aspects, which require a prior mastery of the tool. For this reason, they are left for deepening through other resources, if the reader at any time is dedicated so deeply to the preparation of data with PDI as to need them.



5. Visit our Youtube Channel

Apart from the above-introduced repositories, we have added several videos in our Youtube Channel in order to facilitate the use of some of the main steps for initial transformation steps.

<https://www.youtube.com/@LucentiaResearch>