Data Integration Exercise with Pentaho Data Integration and PostgreSQL

In the data integration exercise, you will use Pentaho Data Integration to transform two data sources and load data into a PostgreSQL table. You will perform transformations to parse date strings, combine fields, and perform validation checks. The two data sources provide new data for the *SSSales* table of the Store Sales data warehouse example. Thus, you need to create the Store Sales tables and sequences and load the sample rows.

You can use the Store Sales tables on a local database with PostgreSQL installed on your PC. The instructions in the exercise demonstrate connection to PostgreSQL on your local machine.

You also need to download the input files (Excel file and Access database file) available in the class website. You will use these input files in the beginning steps of the two job designs that you will create.

This tutorial uses the community edition (CE) of PDI. The latest version (9.1) was installed in March 2021 using SourceForge <https://sourceforge.net/projects/pentaho/>. Figure 1 shows the launch page of the latest version.

A picture containing chart

Description automatically generated

Figure 1: Pentaho Data Integration Welcome Window

To install Pentaho, you should follow the steps below. It is highly recommended that you use the community edition from SourceForge as the instructions in this document follow the community edition interface.

* The latest PDI version in April 2021 on SourceForge is 9.1.0. You should be able to use this version or a newer version to complete the tutorial and assignment.
* You can find the community edition download for version 9.1.0 and other versions at <https://sourceforge.net/projects/pentaho/files/>.
* Unzip the downloaded zip file to any folder.
* Windows users should copy the folder data-integration to the folder C:\Program Files\Pentaho. Mac and Linux users may move the file to any folder.

To ensure that the installation worked, you should launch Pentaho Data Integration.

* Run the file Spoon.bat by double clicking on it. You may want to create a shortcut to the spoon.bat file so starting data integration is easier. If you get a permission error or cannot execute the bat file, you should right click and select “Run as Administrator”. For Mac and Linux users, run the Spoon.sh from terminal (./spoon.sh).
* After you launch Pentaho Data Integration, you will see the Welcome window (Figure 1) and then the Spoon designer (Figure 2).
* Exit Spoon before installing the database driver file in the next part of the instructions.

After you launch Pentaho Data Integration, the Spoon designer is launched at the same time (Figure 2). *Spoon* provides a graphical interface that supports creation of transformations (data flows) and jobs (execution sequences) as well as execution and testing of Pentaho Data Integration processes. Spoon builds jobs and transformations and can save them as database repository and files.

Graphical user interface, application

Description automatically generated

Figure 2: Spoon Opening Window

# 1. Managing Database Connections

Pentaho Data Integration allows you to define connections to multiple databases provided by multiple database vendors (MySQL, Oracle, PostgreSQL, and many more). Pentaho Data Integration installs with the most suitable JDBC drivers for supported databases and its primary interface to databases is through JDBC. Vendors write a driver that matches the JDBC specification and Pentaho Data Integration uses the driver. Unless you require extensive debugging or have other needs, you will not ever need to write your own database driver.

When you define a database connection, the connection information (username, password, port number, and so on) is stored in the Pentaho Enterprise Repository and is available to other users when they connect to the repository. If you are not using the Pentaho Enterprise Repository, the database connection information is stored in the XML file associated with a transformation or job.

Connections that are available for use with a transformation or job are listed under the Database Connectionstep in the explorer View in Spoon.

There are several ways to define a new database connection. You will configure the database connection later in this tutorial.

* In Spoon, under View in the navigation tab, right click Database connections and choose New.
* In Spoon, under View in the navigation tab, right click Database connections and choose New Connection Wizard.
* In the Table input configuration box, click on New.

**Adding a JDBC Driver**

Pentaho Data Integration uses database drivers to provide connections and other operations with databases. Installation of Pentaho Data Integration provides a standard set of database drivers. For PostgreSQL, the installation of Pentaho Data Integration includes a driver for PostgreSQL so you should not need to install a driver. This tutorial used PostgreSQL 12.2 without a need to install an additional database driver. The transformations in the tutorial were also executed using PostgreSQL V13 using the same default database driver.

I have not tested PDI with PostgreSQL with the Mac OS X. I doubt that a database driver download is necessary. If you cannot make a connection to PostgreSQL, you can try installing the jdbc driver for postgresql.

* https://help.pentaho.com/Documentation/9.1/Setup/JDBC\_drivers\_reference
* Copy the driver JAR file to the data-integration/lib folder where the data-integration folder resides inside the folder containing the Pentaho startup files (spoon.sh).

# 2. Overview of Transformations in the Exercise

Spoon is the desktop client component of PDI supporting creation of transformations and jobs. Transformations describe data flows such as reading from a source, transforming data, and loading it into a target database table. Jobs coordinate data integration 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? You can fine more details about PDI in textbook Chapter 14.3.3 (complete textbook) or Chapter 3.3.3 (Data Warehouse Essentials).

A Pentaho transformation executes as a data pipeline with steps connected by directed hops. The output of a prior step flows into the next step as indicated by the hop connecting the steps. Pipeline processing is a well-established processing model amenable to optimization and parallel processing depending on hardware configuration. Steps can execute in parallel, operating on different input records.

During execution of a transformation, the Pentaho processing engine manages a data structure known as the stream. Execution of a step modifies the stream such as by adding new fields in a stream record, sorting records on the stream, or deleting stream records. For example, the Filter step deletes rows on the stream, while the Sort rows step orders records on the stream.

This exercise involves development of two similar transformations shown in Figures 3 and 4. Both transformations process an input file containing rows to insert into the *SSSales* table of the Store Sales data warehouse. The initial input step (Microsoft Excel worksheet or Microsoft Access table) creates stream records with fields corresponding to columns in the *SSSales* table. Most steps in the transformations perform validations to ensure insertion of valid rows in the *SSSales* table. The Filter rows step deletes records with a null value in any field. The Merge join step (Merge Join, Merge Join 2, Merge Join 3, and Merge Join 4 in Figures 3 and 4) combines two streams on a join condition to ensure valid foreign key values. Records not matching on the join condition are deleted from the stream. Merge joins require sorting of records in the same order. In Figure 3, the Merge Join step combines the streams starting with the SSExcelData step (a Microsoft Excel input step) and the *SSTimeDim* step (Table input step). The Add sequence step in both transformations adds a sequence value to the stream, used as the primary key to insert rows in the *SSSales* step (Insert/Update step). A sequence is an object used by Oracle and PostgreSQL to generate unique values for primary keys.

The Access transformation (Figure 4) uses two additional steps (Select values and Split fields) to parse date components. The Access table step has a column with date values that must parsed for the Merge join step. The Merge join step matches on the year, month, and day components of a date value.

Diagram

Description automatically generated

Figure 3: Transformation using Microsoft Excel Input

Diagram

Description automatically generated

Figure 4: Transformation using Microsoft Access Input

To provide guidance about fields in stream records, Pentaho provides menu choices to examine the input and output fields for each step in a transformation. A right click on a step provides a menu with items (Figure 5) for examining the input (Input Fields …) and output (Output fields …) fields on the stream for that step. Selecting Input Fields … for the Sort rows step of the Excel transformation (Figure 3) provides details of the stream input to the step as shown in Figure 6. All fields originate (Step origin) in the SSExcelData step.

Graphical user interface, text, application, email

Description automatically generated

Figure 5: Menu Choices for Each Step

A picture containing text, screenshot, indoor

Description automatically generated

Figure 6: Stream Fields for the Sort rows Step

# 3. Creating your first transformation and loading Excel worksheet

This exercise will step you through building your first transformation with the Spoon client of Pentaho Data Integration introducing common concepts along the way.

Follow the instructions below to create a new transformation.

1. After starting Pentaho Data Integration, you will see the opening window (Figure 1) and the Spoon window (Figure 2).

2. Click  (New File) in the upper left corner of the Spoon window.

3. Select **Transformation** from the list of components (Figure 7) displayed after selecting the **New File** button.

Graphical user interface, application

Description automatically generated

Figure 7: Spoon New File List

Make sure that you have downloaded the Excel input file from the class website. You need to know the location of this file in Step 4 below.

Step 1 – In the View tab, right click the new transformation 1 and select “settings…”

Step 2 – Set the Transformation name for the new transformation as: SSTORETEST and click OK.

Step 3 – Save the transformation following **File → Save.** You will see the empty transformation window in the Spoon (Figure 8).

Graphical user interface

Description automatically generated

Figure 8: Empty Transformation Window

Step 4 – Create the Excel Input step:

* Under the Design tab, expand the Input step (Figure 9).

Graphical user interface, application

Description automatically generated

Figure 9: List of Input Steps with Microsoft Excel Input step in the Transformation Design Window

* Select and drag a **Microsoft Excel Input** step into the canvas on the right.
* Double Click on the **Microsoft Excel Input** step. The edit properties dialog box (Figure 10) associated with the **Microsoft Excel Input** step appears. In this dialog box, you specify the properties related to a particular step.

Graphical user interface, text, application, email

Description automatically generated

Figure 10: Files Window for Microsoft Excel Input Property Editing

* Set name for the Excel Input as **SSExcelData** and specify the Excel data source path in the **Files** tab.
* In the tab named **Files**, click the button “Browse…” and locate the Excel file that you downloaded from the class website. Then, Click “Add” to add the file to the selected files area.
* In the tab named **Sheets**, click the button “**Get sheetname(s)…**”. There will appear an **Enter List** (Figure 11) to choose sheets. Select **Sheet 1**, press “**>**” to move it into the right area. Click **OK**.
* In the tab names **Fields,** click on **“Get fields from header row…”** You need to change the data types, length, and precision as the specification in Figure 12.

Graphical user interface, application

Description automatically generated

Figure 11: Sheet Specification Window

Graphical user interface, application, table, Excel

Description automatically generated

Figure 12: Fields Window for Microsoft Excel Input Property Editing

* To ensure that the Excel input can read rows from the associated worksheet, select the Preview rows button. Figure 13 shows the preview of rows. Close the window when finished previewing rows.
* Click **OK** at the bottom of the window. The input icon will change to the icon displayed in Figure 14.

Graphical user interface, table

Description automatically generated

Figure 13: Preview Rows Window

Graphical user interface, application, qr code

Description automatically generated

Figure 14: SSExcelData Icon

Step 5 – In this part of the tutorial, you will add constraint checking for null values in stream records obtained from the Excel data source. The rows from the Excel data source were added to the stream (data maintained in a Pentaho transformation) in the output of the first step.

* Add a Filter Rows step to your transformation. Under the **Design** table, go to **Flow** → **Filter Rows**.
* Create a “hop” between the **SSExcelSource** (Excel file input) step and the **Filter Rows** step. Hops are used to describe the flow of data in your transformation. To create the hop, click the **SSExcel Source** (Excel file input) step, then press the <**SHIFT**> key down and draw a line to the Filter Rows step (Figure 15).

Text

Description automatically generated with medium confidence

Figure 15: Hop connecting an Excel Input Step Connected to a Filter Rows Step

* Alternatively, you can draw hops by hovering over a step until the hover menu (Figure 16) appears. Drag the hop painter icon from the source step to your target step.

Diagram

Description automatically generated with medium confidence

Figure 16: Hover Menu

* Double-click the **Filter Rows** step. The **Filter Rows** edit properties dialog box appears (Figure 17).

Graphical user interface, text, application, email

Description automatically generated

Figure 17: Property Edit Window of Filter Step

* The **Step Name** field is **Filter rows**.
* Under **The condition**, click <field>. A dialog box that contains the fields you can use to create your condition appears.
* In the **Fields**: dialog box (Figure 18) select **SalesUnits** and click **OK**.
* Click on the comparison operator (Figure 19) (set to = by default) and select the **IS NOT NULL** function and click **OK**.
* Click the button . A new condition row appears with **null = [ ]** as a default.
* Click on the expression and add constraints for the next column similarly to what you did for “**SalesUnits**”
* Click on **UP**. This will allow you to see both conditions connected by AND.
* Click the button again. Another new condition row appears with **null = [ ]** as a default.
* Keeping repeating these steps for all fields.
* The final view of filter conditions is shown by Figure 20.
* Save your transformation.

Graphical user interface

Description automatically generated

Figure 18: Condition Fields Selection Window

Graphical user interface

Description automatically generated

Figure 19: Comparison Operator List

Graphical user interface, text, application, email

Description automatically generated

Figure 20: Filter Conditions Window

Step 6 – Create a step to sort the result of the Filter Rows step.

* Under the **Design** tab, expand the contents of the **Transform** category.
* Click and drag a **Sort Rows** step into your transformation; create a hop between the **Filter rows** and Sort Rows steps. Select **Result is TRUE** in the filter results selection list (Figure 21).

Diagram

Description automatically generated with medium confidence

Figure 21: Filter Results Selection List

* Double-click the **Sort Rows** step to open its edit properties dialog box (Figure 22). Click “**Get Fields**” to obtain the fields. Delete other fields except the Day, Month and Year fields. Then click Ok.

Graphical user interface, application

Description automatically generated

Figure 22: Property Edit Window of Sort Rows Step

# 4. Lookup Columns from the PostgreSQL tables

This part of the tutorial involves looking up the date from the *SSTimeDim* table to check the validity of dates in the Excel data source. In addition, you will lookup primary key columns from other PostgreSQL tables to ensure loaded data does not contain invalid foreign keys.

Step 1 – Access the *SSTimeDim* table from PostgreSQL database.

* Under the **Design** tab, expand the contents of the **Input** step.
* Click and drag a **Table Input** step into your transformation.
* Double-click the Table Input step to open its edit properties dialog box (Figure 23).
* Rename your Table Input step to *SSTimeDim*.

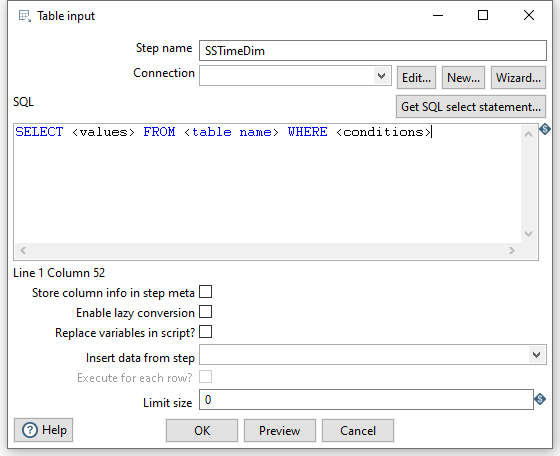


Figure 23: Property Edit Window of Table Input Step

* Click “**New…**” next to the connection field. You must create a connection to the database. The Database connection dialog box appears.
* Provide the settings for connecting to the database as shown in Figure 24. You can choose any name for the connection.
* **IMPORTANT**: Before setting the connection information, you should ensure that the database is created in PostgreSQL along with the tables and the records inserted into those tables. Here are the details to connect to the PostgreSQL 12.2. The Database Name and the Username are the one is the one that you create in PostgreSQL. The full value for database connection is given in the remainder of step 1.

Graphical user interface, application

Description automatically generated

Figure 24: Database Connection Window

* Connection Name: PostgreSQL12.2 (You can use another name if you want)

Connection Type: PostgreSQL

Host Name: localhost

Database Name: StoreSales (Use the database name that you used.)

Port Number: 5432 is the default port number. I have two PostgreSQL database servers install with 5432 for V12.2 and 5433 for V13.2. To check the port used by your installed PostgreSQL database server, right click on the server in pgAdmin. Then click **Properties…** You can find the port in the Connection tab. You will not be able to connect to your PostgreSQL database server if you use an incorrect port number.

Username: postgres (default administrative user name)

Password: <blank> (if password was not specified or use password specified.)

Access: Native (JDBC)

Note: The username “postgres” is the default administrative user after installation of PostgreSQL. For the password, you need to use the password that you specified if any. In this example, the password for the postgres user was omitted during PostgreSQL installation. The Host Name and port number are the default values for Windows 10 installation of PostgreSQL version 12.

* Click “Test” to test the connection. Then success test result is shown by Figure 25.

Graphical user interface, text, application

Description automatically generated

Figure 25: Database Connection Test

* Type in “SELECT \* FROM SSTimeDim” in the SQL section (Figure 26). You can click the **Preview** button to view the database. Click Ok, to exit the Database Connection dialog box.

Graphical user interface, text, application, email

Description automatically generated

Figure 26: SQL Edit Section in Property Window of Table Input Step

* Add another sort rows component **Sort rows 2**, and a hop connecting the *SSTimeDim* step. In the field specification (Figure 27), delete other fields except TIMEDAY, TIMEMOHTH, TIMEYEAR fields.

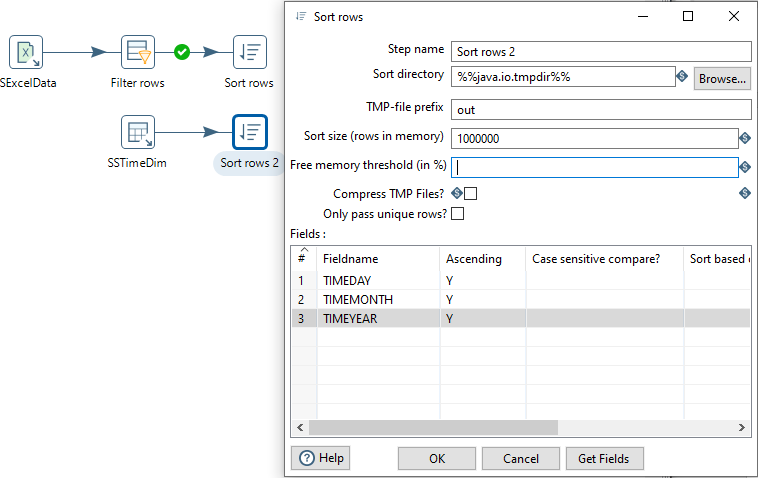


Figure 27: Property Edit Window of Sort Rows 2 Step

* Under the **Design** tab, expand the contents of the **Joins** step.
* Click and drag a **Merge Join** step into your transformation; create a hop between the **Sort rows, Sort rows 2** and **Merge Join** steps (Figure 28).

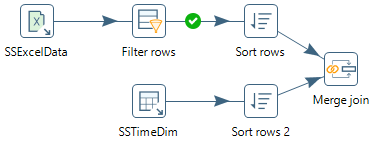


Figure 28: Two Sort Rows Steps Connected to Merge Join Step

* Double-click the Merge Join step to specify its properties (Figure 29). Set **First step** as **Sort rows**, **Second step** as **Sort rows 2**, and **Join Type** as **INNER**. Click both of the “**Get key fields**” at left and right to get the possible fields to join. In the left table, delete other fields except Day, Month and Year fields. In the right table, delete other fields except *TIMEDAY*, *TIMEMONTH*, and *TIMEYEAR* fields. Then click OK.

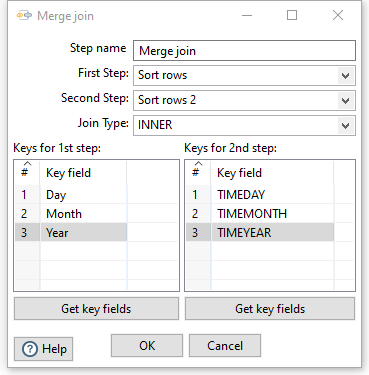


Figure 29: Property Edit Window of Merge Join Step

* Now, we have finished inner join between Excel input and *SSTimeDim* table.

Step 2 – Inner join the *SSItem*, *SSCustomer*, and *SSStore* tables.

Like getting data from the *SSTimeDim* table in the previous section, inner joining these tables requires **Table Input** components. First, you should set the connection and SELECT statement for the *SSItem* table. Note that these tables should exist in your PostgreSQL schema before these steps.

* Drag and drop the **Table Input 2** into the design pane.
* Double click on the newly created component to open its Basic Settings pane. Specify the connection as shown in previous figure.
* Use “SSItem” as the Table Name value and “SELECT \* FROM SSItem” as the Query value.
* Create two **sort rows** components: **Sort rows 3** and **Sort rows 4**, connecting **Merge Join** and **SSItem** respectively. See the field to be sorted as: **ItemID** and **ITEMID** respectively.
* Drag and drop the **Merge Join 2** into the design pane. Connect **Sort rows 3** and **Sort rows 4** to **Merge Join 2**. Set the field to be joined as **ItemID** and **ITEMID**.
* Figure 30 shows all steps and hops to the Merge join 2 step.

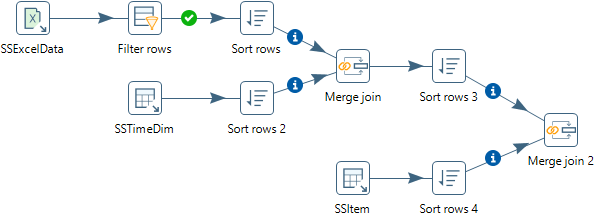


Figure 30: Transformation Design Showing Steps and Hops to the Merge Join 2 Step

Step 3 – Inner join the tables.

* Inner join the tables named *SSCustomer* and *SSStore* in your transformation using the same method described previously.
* For the *SSCustomer* step, connect the *CustID* (from Excel file) and CUSTID (from Database) fields.
* For the SSStore step, connect the *StoreID* (from Excel file) and STOREID (from Database) fields.
* Figure 31 shows all steps and hops after the Merge join 4 step.

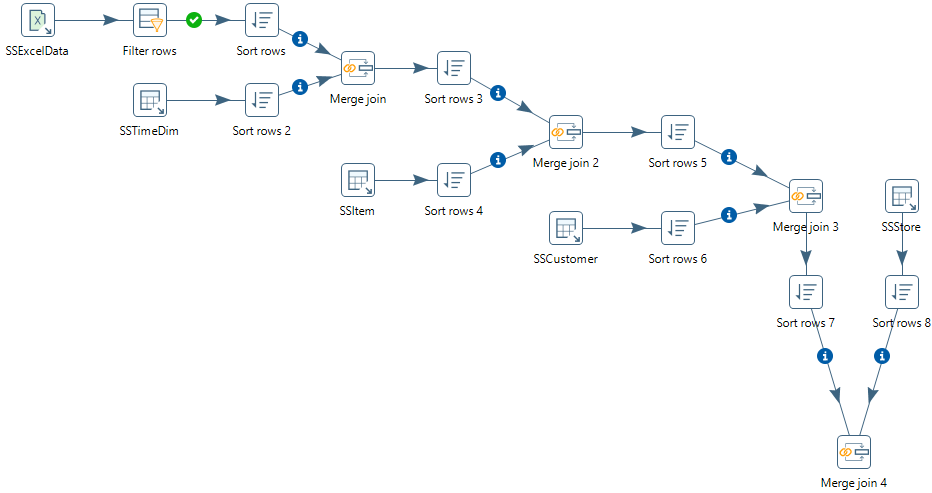


Figure 31: Transformation Design Showing Steps and Hops after the Merge Join 4 Step

Step 4 – Create and connect an Add Sequence step to generate values for the *SalesNo* column.

* Under the **Design** tab, expand the contents of the **Transform** step.
* Click and drag an **Add sequence** step into your transformation; create a hop between the **Merge Join 4** and **Add Sequence** steps (Figure 32). To create the hop, click the **Merge Join 4** step, then hold the <**SHIFT**> key down and draw a line to the **Add Sequence** step.
* Double click on the newly created component to open its Basic Settings pane.
* Set **SalesNo** as the name of value. Check the box for use DB to get sequence. Select the **connection** as **PostgreSQL12.2.** Set **SSSalesNoSeq** as sequence name (Figure 33).

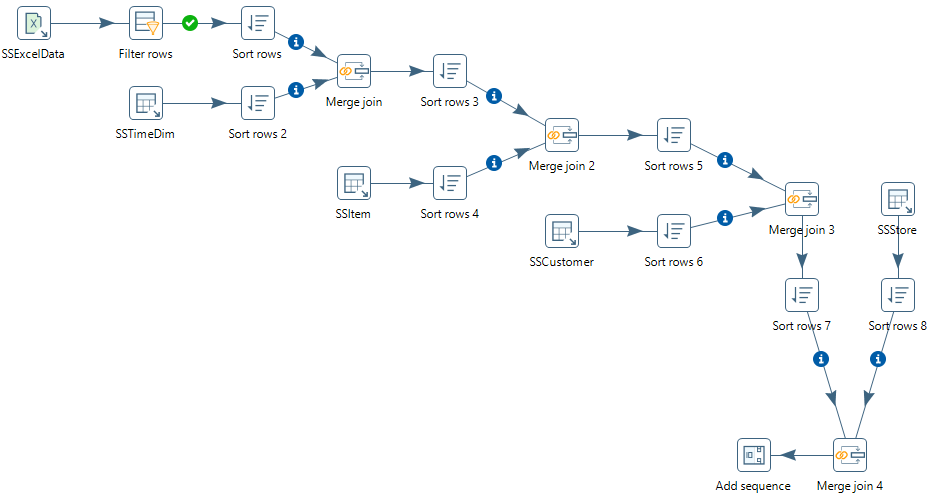


Figure 32: Transformation Design Showing Steps and Hops after the Add Sequence Step

A screenshot of a cell phone

Description automatically generated

Figure 33: Property Edit Window of Add sequence Step

# 5. Insert data into the SSSales table

* Under the **Design** tab, expand the contents of the **Output** step.
* Click and drag an **Insert/Update** step into your transformation; create a hop between the **Add sequence** and **Insert/Update** steps. Figure 34 shows the Insert/Update step (**SSSales**) connected to Add sequence step.

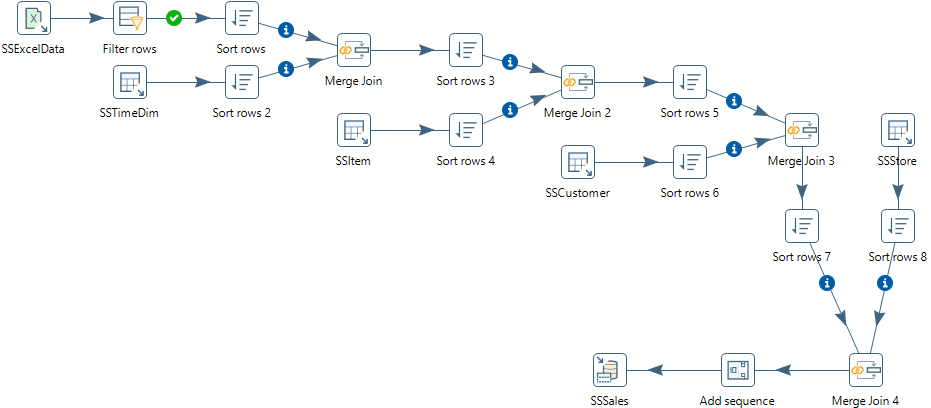


Figure 34: Connect Insert/Update Step to the Add Sequence Step

* Double click the **Insert/Update** step, to specify its properties (Figure 31). Set the **step name** as **SSSales**. Select the **connection** as **PostgreSQL12.2**. Type in the **Target table** as **SSSales** or click the **Browse** button and select the table from the list. Do not click the “**Get fields**” button. Instead, select SalesNo from the two sources and set the comparator to **=**. The final window should look like Figure 35.
* Click the button “**Get Updated fields**” and then click on “**Edit mapping**” button to edit mapping. The mapping edit window is shown in Figure 36. Select the fields named **SalesUnits**, **SalesDollar**, **SaleCost**, **CustID,** **StoreID**, **ItemID** **TIMENO** and **SalesNo** into the **mappings** field. Pentaho will automatically match the corresponding name in the Target field. You may need to choose **SalesNo** stream field has matched with **SALESNO** column. Then click **OK**.

Table

Description automatically generated

Figure 35: Property Edit Window of Insert/Update Step

Graphical user interface

Description automatically generated

Figure 36: Mapping Edit Window

* Check the button “Don’t perform any updates:”. The final view of the **SSSales** step will look like Figure 37.

Table

Description automatically generated

Figure 37: Final view of the SSSales step

* Select the **SSSales** step and run a preview by clicking on . In the transformation debug dialog, click on **Quick Launch** (Figure 38).
* The Examine preview data window is displayed in Figure 39.

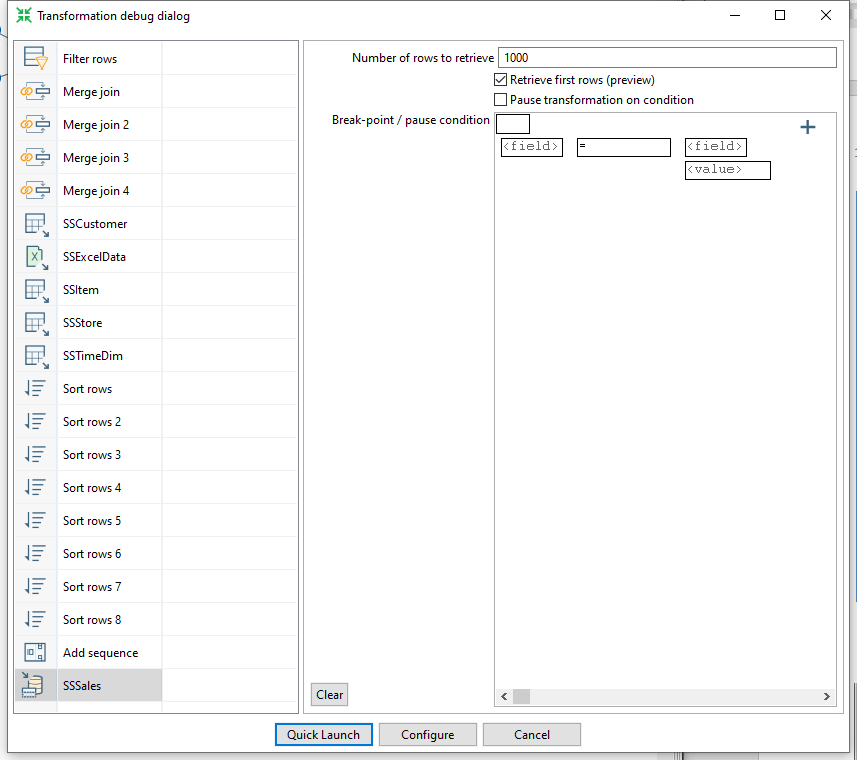


Figure 38: Transformation Debug Dialog

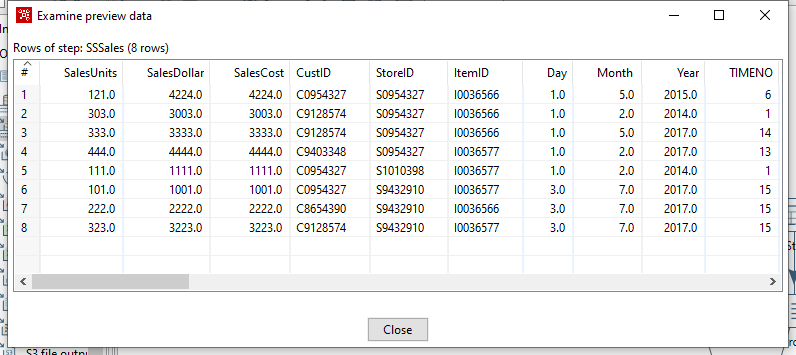


Figure 39: Examine Preview Data Window

* To examine the details of each step, you should examine the Execution Results window below the design pane. The Step Metrics tab (Figure 40) shows details about the execution of each step. You should verify that the **SSSales** step has 8 output rows.

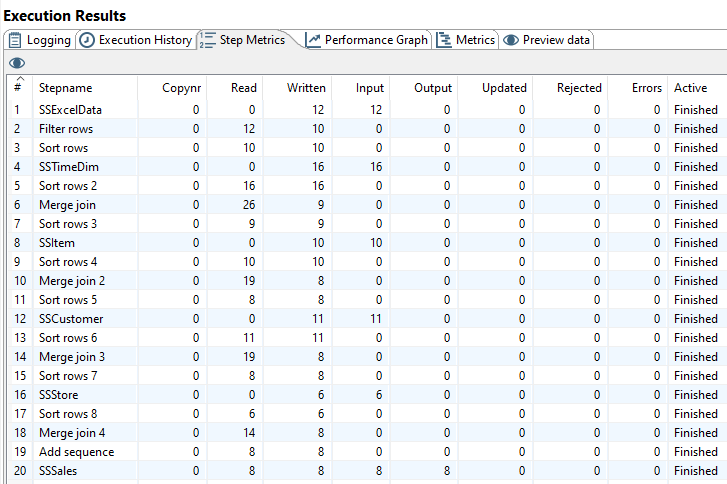


Figure 40: Step Metrics in the Execution Result Window

* Each step in the transformation should have a check mark indicating execution as shown in Figure 41.
* Connect to your PostgreSQL account (on your PC) so you can verify the number of rows in the *SSSales* table. You should see 200 rows with 8 new rows added to the 192 rows in the PostgreSQL *SSSales* table.
* If you do not see the extra rows, the PostgreSQL output component had a failure. To see the error, check the Logging and Step Metrics tabs of the **Execution Results** window.

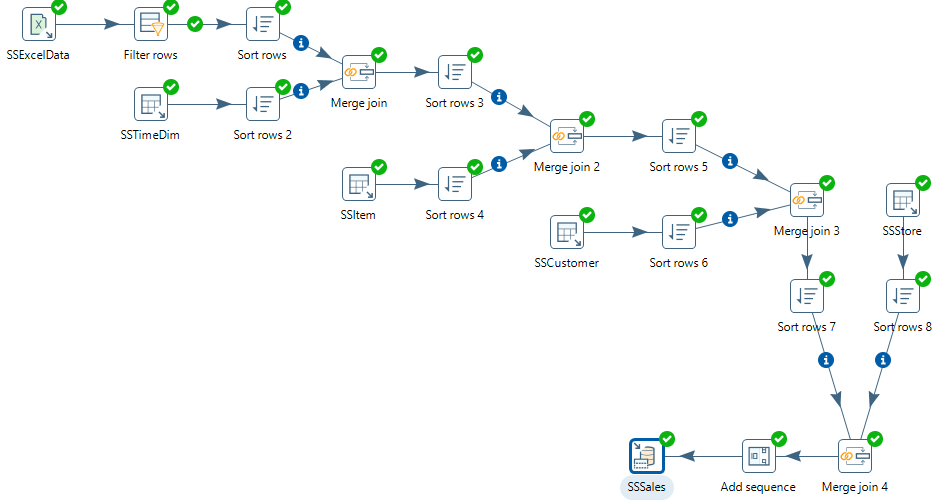


Figure 41: Transformation Design with Check Marks for Each Step

# 6. Load second data source from Access

The next part of the exercise involves creation of a new transformation to process the Access data source. Make sure that you have downloaded the Access database file from the class website and noted its location on your computer. Create a new transformation using **File → New → Transformation** with name “SSStoreTestAccess”. Use **File** **→** **Save As …** to save the transformation file as “SSStoreTestAccess” to a folder of your choice. Then, you will begin by loading the rows from a table in the Access database.

Step 1- Add the Access Input Step

* Under the Design tab, expand the Input step. Figure 42 shows the Design table and input step.
* Select and drag a **Microsoft Access Input** step onto the canvas on the right.
* Double Click on the **Microsoft Access Input**. The edit properties dialog box associated with the **Microsoft Access Input** step appears (Figure 43). In this dialog box, you specify the properties related to this step.

Graphical user interface, text, application, email

Description automatically generated

Figure 42: New Microsoft Access Input Step

Graphical user interface, text, application, email

Description automatically generated

Figure 43: Property Edit Window of Microsoft Access Input Step

* Set name for the Access Input as **Sales** and specify the Excel data source path in the **Files** tab.
* In the tab named **Content**, click the button “**Get tables**” of **table** section. There will appear a window (Figure 44). Select **Sales** as the table name, click **OK**.

Graphical user interface, text, application, email

Description automatically generated

Figure 44: Table Selection Window

* In the tab named **Fields**, click the button “**Get fields**”. There will appear a list (Figure 45) showing the fields in the table named **Sales**.

Graphical user interface, application, table

Description automatically generated with medium confidence

Figure 45: Fields Window for Microsoft Access Input Property Editing

* Click the button “**Preview rows**” to preview the database (Figure 46). When asked for the number of rows type 12 and click OK.

Graphical user interface, table

Description automatically generated

Figure 46: Examine Preview Data Window

* Click **OK** at the bottom of the window. The input icon will change to the shape shown by Figure 47.



Figure 47: Sales Step Icon

Step 2 –You will add constraint checking for null values using the Filter Rows step.

* Add a Filter Rows step to your transformation. Under the **Design** table, go to **Flow** → **Filter Rows**.
* Create a hop between the **Sales** (Access file input) step and the **Filter Rows** step. Hops are used to describe the flow of data in your transformation. To create the hop, click the **Sales** (Access file input) step, then press the <**SHIFT**> key down and draw a line to the Filter Rows step. The hop should be the main output of the **Sales** step. Figure 48 shows the transformation window after adding the new step and hop.

Graphical user interface, text, application

Description automatically generated

Figure 48: Access Input Step and Filter Step in Spoon

* 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.
* Double-click the **Filter Rows** step. The **Filter Rows** edit properties dialog box appears.
* In the **Step Name** field type, **Filter rows**.
* The configuration of this step is like the previous Excel transformation.
* The final view of filter conditions is shown in Figure 49. Save the transformation before adding new steps.

Graphical user interface, application

Description automatically generated

Figure 49: Filter Conditions Window

# 7. Separate SalesDay fields into Day, Month, Year fields

In this part of the tutorial, you will use the Select Values step to change the format of the myDate field and the Split Fields step to parse the field into date components.

* Under the **Design** tab, expand the contents of the **Transform** step.
* Click and drag a **Select values** step into your transformation.
* Create a “hop” between the **Filter rows** step and the **Select values** step (Figure 50). Select **Result is TRUE** in the filter results selection list.

Diagram

Description automatically generated

Figure 50: True Filter Results Connected to Select Values Step

* Double-click the Select values step to open its edit properties dialog box.
* In the tab named Meta-data, click the button “**Get fields to change**”, to get the fields to change, which is shown by Figure 51. Change the **Type** of field **myDate** as **String**, change its **Format** to dd-MM-yyyy. Click **OK**.

Table

Description automatically generated with medium confidence

Figure 51: Meta-data Tab of Select Values Property Edit Window

* Under the **Design** tab, expand the contents of the **Transform** step.
* Click and drag a **Split fields** step into your transformation (Figure 52).
* Create a “hop” between the **Select values** step and the **Split fields** step. The hop should be the main output of the previous step.

Diagram, text

Description automatically generated with medium confidence

Figure 52: Create Split Fields with Hop between Steps

* Double-click the **Split fields** step to open its edit properties dialog box (Figure 53).
* Select **myDate** in the **Field to split**, type “**-**” as the **Delimiter**. Type in **Year, Month** and **Day** in the Column named **New field**, and set their **Type** as **Integer**. Click Ok when finished.

Graphical user interface, table

Description automatically generated

Figure 53: Property Edit Window of Field Splitter Step

* Select the Split fields step in the canvas and click  , to preview this transform (Figure 54). Make sure that Split Fields step is selected from the left side panel of the transformation debug dialog and click on “**Quick Launch**” button.

Table

Description automatically generated

Figure 54: Examine Preview Data Window

# 8. Lookup Columns from the PostgreSQL tables

This part of the exercise involves looking up the date from the *SSTimeDim* table to check the validity of dates in the Access data source. In addition, you will lookup primary key columns from other PostgreSQL tables to ensure loaded data does not contain invalid foreign keys. This part of the exercise resembles details in Section 4.

Step 1 – Access the *SSTimeDim* table from PostgreSQL database.

* Under the **Design** tab, expand the contents of the **Input** step.
* Click and drag a **Table Input** step into your transformation.
* Double-click the Table Input step to open its edit properties dialog box.
* Rename your Table Input step to *SSTimeDim*.
* For the Connection field, select PostgreSQL12.2 if it is available in the connection list. Otherwise, click “**New**” next to the connection field. Provide the settings for connecting to the database as shown in the Figure 24.
* Connection Name: PostgreSQL12.2 (You can use another name if you want)

Connection Type: PostgreSQL

Host Name: localhost

Database Name: StoreSales (unless you used a different name in PostgreSQL)

Port Number: 5432

Username: postgres (default administrative user for PostgreSQL)

Password: <blank> (or the password used when you installed PostgreSQL)

Access: Native (JDBC)

* Click “**Test**”, to test the connection.
* Type in “SELECT \* FROM SSTimeDim” in the SQL section. You can click the **Preview** button to view the database. Click Ok, to exit the Database Connection dialog box.
* Under the **Design** tab, expand the contents of the **Transform** step.
* Click and drag a **Sort Rows** step into your transformation; create a hop between the **Split fields** and **Sort Rows** steps.
* Double-click the **Sort Rows** step to open its edit properties dialog box. Click “**Get fields**” to obtain the fields. Delete other fields except the Day, Month and Year fields. Then click Ok.
* Add one more sort rows component **Sort rows 2**, and a hop connecting the *SSTimeDim* step. In the field specification, delete other fields except *TIMEDAY*, *TIMEMOHTH*, *TIMEYEAR* fields.
* Under the **Design** tab, expand the contents of the **Join** step.
* Click and drag a **Merge Join** step into your transformation; create a hop between the **Sort rows, Sort rows 2** and **Merge Join** steps.
* Double-click the Merge Join step to specify its properties. Set **First step** as **Sort rows**, **Second step** as **Sort rows 2**, and **Join Type** as **INNER**. Click both of the “**Get key fields**” at left and right to get the possible fields to join. In the left table, delete other fields except Day, Month and Year fields. In the right table, delete other fields except *TIMEDAY*, *TIMEMONTH*, and *TIMEYEAR* fields. Make sure that the steps are in the same order (day, month, year) in each step part. Then click OK.
* Now, we have finished inner join between the Access table and *SSTimeDim* table.
* Figure 55 shows the transformation design with all steps and hops to the Merge join step.

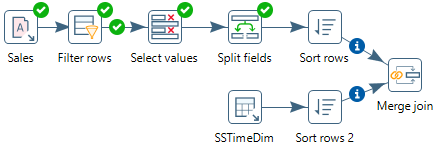


Figure 55: Transformation Design with Steps and Hops to the Merge Join Step

Step 2 – Inner join *SSItem*, *SSCustomer*, and *SSStore* to Access table.

* Inner join the tables named *SSItem*, *SSCustomer*, and *SSStore* in your transformation using the same method described before.
* For *SSItem* step, connect *ItemID* (from Excel file) and *ITEMID* (from Database) fields.
* For *SSCustomer* step, connect *CustID* (from Excel file) and *CUSTID* (from Database) fields.
* For *SSStore* step, connect *StoreID* (from Excel file) and *STOREID* (from Database) fields.
* Figure 56 shows the transformation design for steps and hops to the Merge join 4 step.

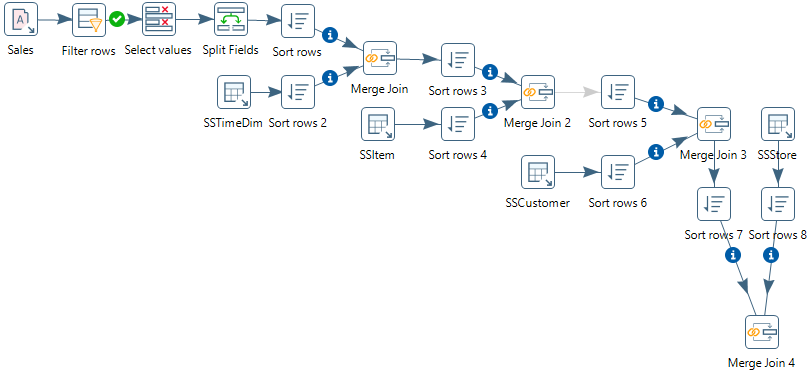


Figure 56: Transformation Design with Steps and Hops to the Merge Join 2 Step

Step 3 – Add SalesNo column.

* Under the **Design** tab, expand the contents of the **Transform** step.
* Click and drag **Add sequence** step into your transformation; create a hop between the **Merge Join 4** and **Add Sequence** steps (Figure 57).
* Double click on the newly created component to open its Basic Settings pane.
* Set **SalesNo** as the name of value. Check the box for use DB to get sequence. Select the **connection** as **PostgreSQL12.2.** Set **SSSalesNoSeq** as sequence name (Figure 58).

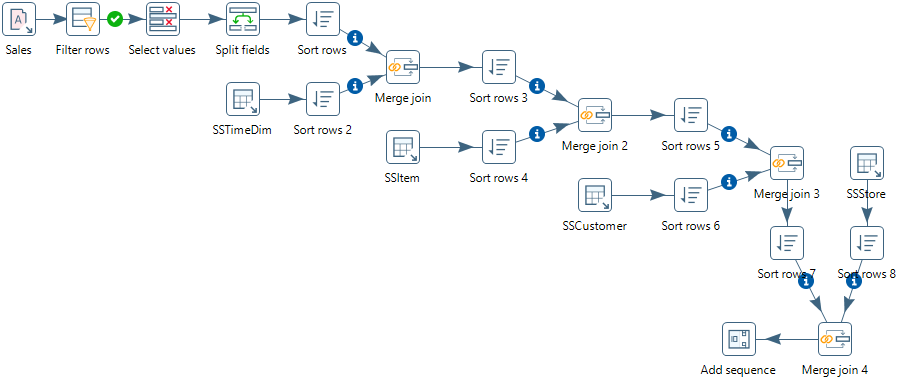


Figure 57: Transformation Design with Steps and Hops to the Add Sequence Step

Graphical user interface, text, application, email

Description automatically generated

Figure 58: Property Edit Window of Add sequence step

# 9. Insert rows into the SSSales table

* Under the **Design** tab, expand the contents of the **Output** step.
* Click and drag an **Insert/Update** step into your transformation; create a hop between the **Add sequence** and **Insert/Update** steps. Figure 59 shows the connection in the transformation design pane.
* Double click the **Insert/Update** component, to specify its properties. Set the **step name** as **SSSales**. Select the **connection** as **PostgreSQL12.2**. Type in the **Target table** as **SSSales**. Do not click the “**Get fields**” button. Instead, select SalesNo from the two sources and set the comparator to **=**. Figure 56 shows the window with the lookup values in the middle part.
* Click the button “**Get Updated fields**” and then click on “**Edit mapping**” button to edit mapping. Select the fields named **SalesUnits**, **SalesDollar**, **SaleCost**, **CustID,** **StoreID**, **ItemID** **TIMENO** and **SalesNo** into the **mappings** field. Pentaho will automatically match the corresponding name in the Target field. Only **SalesNo** column must be manually matched with the **SALESNO** field. Then click **OK**. Figure 60 shows the final window.

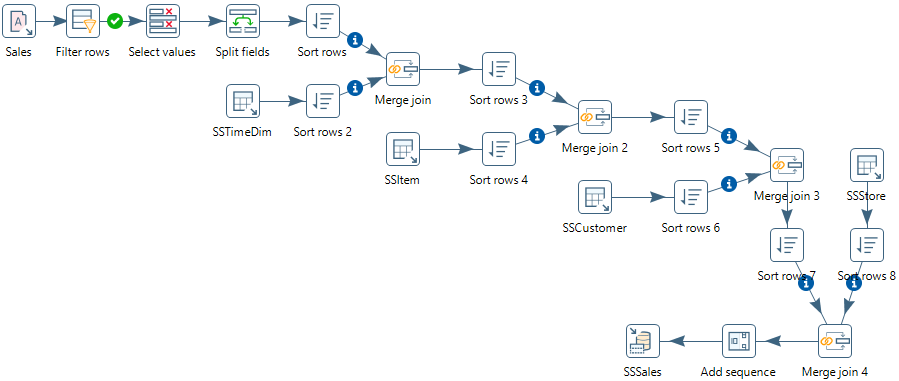


Figure 59: Connect Insert/Update Step to Add Sequence Step

Table

Description automatically generated

Figure 60: Insert/Update Step Window

* Select the **SSSales** step and run a preview by clicking on . In the transformation debug dialog click on **Quick Launch** button. Figure 61 shows the result rows added to the *SSSales* table after execution of the step. The Step Metrics tab (Figure 62) shows that 8 rows were inserted into the *SSSales* table in the Output column.
* Connect to your PostgreSQL account (on your PC) so you can verify the number of rows in the *SSSales* table. You should see 208 rows with 8 new rows added to the 200 rows existing after the Excel transformation execution (192 original rows and 8 rows from the Excel transformation).

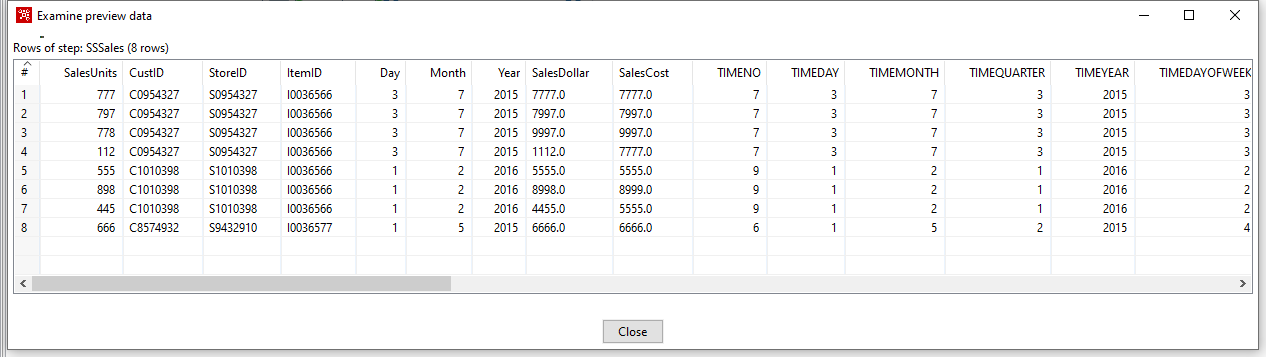


Figure 61: Preview Data for the SSSales Step in the Access Transformation

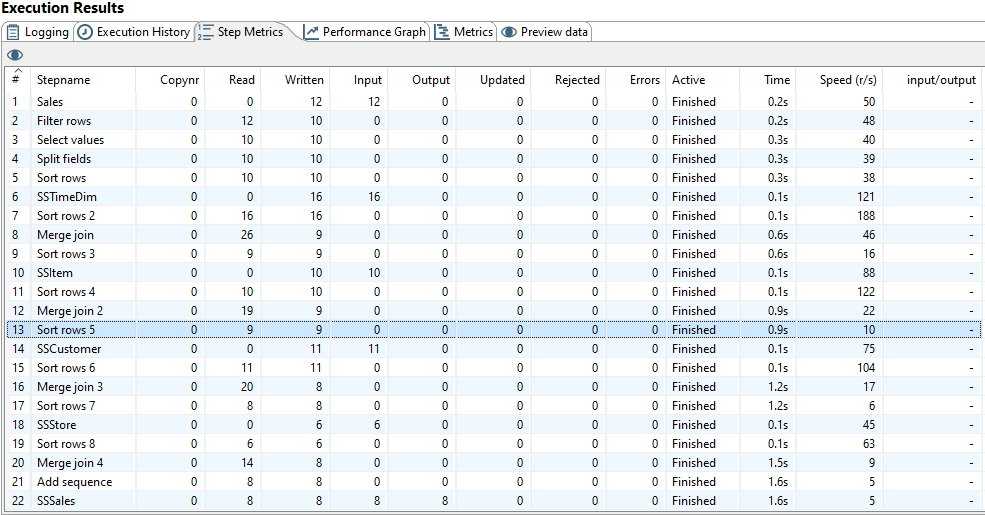


Figure 62: Step Metrics in the Execution Result Window for the Access Transformation