

MINGGU #4

(PDI ADVANCE CONTROL)

DESKRIPSI TEMA

Mahasiswa mampu memiliki pengetahuan dan mampu memanfaatkan fitur advanced control dan process execution monitoring pada aplikasi Pentaho Data Integration (PDI) serta memahami cara menghubungkan dan menyimpan data pada data mart menggunakan database MySQL

CAPAIAN PEMBELAJARAN MATA KULIAH (SUB-CPMK)

CPMK-3 Sub-CPMK-4

Mahasiswa mampu mengimplementasikan proses ETL sederhana dengan sumber data dari database dengan **menerapkan** konsep Job and Transformation PDI-C3.

1. PDI Logging setup
2. Debugging Tips for Transformations and Jobs
3. Implementing the Error Handling Functionality
4. Getting data from relational databases

ALAT PENDUKUNG PRAKTIKUM

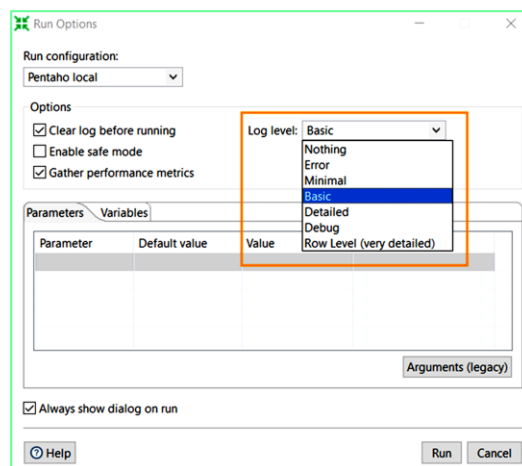
1. Windows Operating System
2. Java Runtime Environment (JRE) & Java Development Kit (JDK) 8.0 or above version (*installed*)
3. Pentaho Data Integration tools (PDI) release 7.1 or above release

LANGKAH-LANGKAH PRAKTIKUM

1. PDI Logging setup

PDI logs all of the executions of a transformation. By default, the level of the logging details is basic, but there are seven possible levels of logging, ranging from Nothing at all to Rowlevel (very detailed), which is the most detailed level of logging. You can change the level of logging as follows:

- a. If you will run a transformation, in the Execute a transformation window, before clicking on Run, select the proper option:

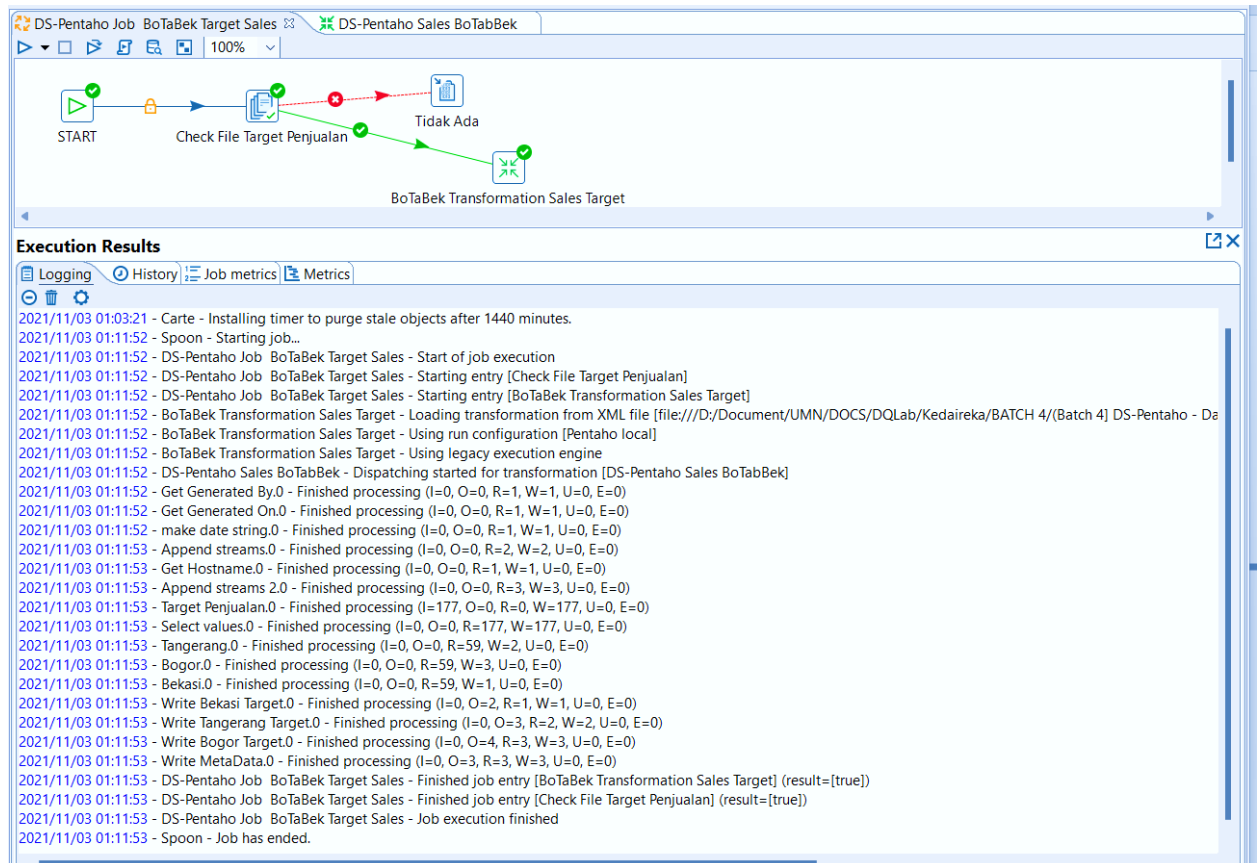


- b. Levels of Logging

In PDI there are various possibilities to set different levels of verbosity logging, of which there are 7 levels depending on the requirements required:

Log Level	Description
Nothing	Do not record any logging output.
Error	Only show errors.
Minimal	Only use minimal logging.
Basic	This is the default level.
Detailed	Give detailed logging output.
Debug	For debugging purposes, very detailed output.
Row Level	Logging at a row level. This will generate a lot of log data.

- c. The following below is an example when using logging level = Basic, the results are displayed on the Logging-tab as follows:

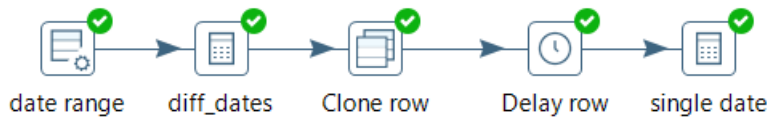


The screenshot displays the Pentaho Data Studio interface. At the top, there are tabs for 'DS-Pentaho Job', 'BoTaBek Target Sales', and 'DS-Pentaho Sales BoTabBek'. Below the tabs, a job design canvas shows a flow starting from a 'START' node, followed by a 'Check File Target Penjualan' node, then a 'Tidak Ada' node, and finally a 'BoTaBek Transformation Sales Target' node. The 'Execution Results' panel at the bottom is active, showing a detailed log of the job execution. The log includes timestamps and descriptions of various steps, such as 'Installing timer to purge stale objects after 1440 minutes', 'Starting job...', 'Start of job execution', 'Starting entry [Check File Target Penjualan]', 'Starting entry [BoTaBek Transformation Sales Target]', 'Loading transformation from XML file', 'Using run configuration [Pentaho local]', 'Using legacy execution engine', 'Dispatching started for transformation [DS-Pentaho Sales BoTabBek]', and 'Finished processing' for various data sources like 'Get Generated By.0', 'Get Generated On.0', 'make date string.0', 'Append streams.0', 'Get Hostname.0', 'Append streams 2.0', 'Target Penjualan.0', 'Select values.0', 'Tangerang.0', 'Bogor.0', 'Bekasi.0', 'Write Bekasi Target.0', 'Write Tangerang Target.0', 'Write Bogor Target.0', and 'Write MetaData.0'. The log concludes with 'Finished job entry [BoTaBek Transformation Sales Target] (result=[true])', 'Finished job entry [Check File Target Penjualan] (result=[true])', and 'Job execution finished'.

2. Debugging Tips for Transformations and Jobs

- ▶ So far, you have learned some basics about working with Spoon during the design process. Now you will continue learning about interacting with the tool.
- ▶ First, we will create a Transformation, aiming to learn some new useful steps. After that, we will adapt that Transformation for inspecting the data as it is being created.
- ▶ As you progress, feel free to preview the data that is being generated, even if you're not told to do so. This will help you understand what is going on. Testing each step as you move forward makes it easier to debug and craft a functional Transformation.

- Let's start with the creation of the Transformation. The objective is to generate a dataset with all data in between a given range of dates:



- Create a new Transformation.
- From the Input group of steps, drag to the canvas the Generate Rows step, and configure it as shown:

Generate Rows

Step name: date range

Limit: 1

Never stop generating rows: ☐

Interval in ms (delay): 5000

Current row time field name: now

Previous row time field name: FiveSecondsAgo

Fields:

#	Name	Type	Format	L...	P...	C...	D...	G...	Value	S...
1	start_date	Date	yyyy-MM-dd						2022-01-01	N
2	end_date	Date	yyyy-MM-dd						2027-12-31	N

**** *Configuring a Generate Rows step Note that you have to change the default value for the Limit textbox, from 10 to 1.*

- Close the window.
- From the Transform category of steps, add the Calculator step, and create a hop that goes from the Generate Rows step to this one.
- Double-click on the Calculator step and add the field named diff_dates as the difference between end_date and start_date. That is, configure it exactly the same way as you did in the previous section.
- Run a preview. You should see a single row with three fields: the start date, the end date, and a field with the number of days between both.

Generate Rows

Step name: date range

Limit: 1

Never stop generating rows: ☐

Interval in ms (delay): 5000

Current row time field name: now

Previous row time field name: FiveSecondsAgo

Fields:

#	Name	Type	Format	Value	Set empty string?
1	start_date	Date	yyyy-MM-dd	2022-01-01	N
2	end_date	Date	yyyy-MM-dd	2027-12-31	N

g.

Calculator

Step name: diff_dates

Fields:

#	New field	Calculation	Field A	Field B	Value ...	Remo...
1	diff_dates	Date A - Date B (in days)	end_date	start_date	Integer	N

Execution Results

Logging Execution History

First rows Last rows Off

#	start_date	end_date
1	2022-01-01	2027-12-31

- Now add the Clone row step. You will find it inside the Utility group of steps.
- Create a hop from the Calculator step towards this new step.
- Edit the Clone row step. Select the Nr clone in field? option to enable the Nr Clone field textbox. In this textbox, type diff_dates.
- Now select the Add clone num to output? option to enable the Clone num field textbox. In this textbox, type delta.
- Run a preview. You should see the following 1K of records:

Clone row

Step name: Clone row

Nr clones: 0

Nr clone in field? ☒

Nr Clone field: diff_dates

Output fields:

Add clone flag ☐

Clone flag field:

Add clone num ☒

Clone num field: delta

Examine preview data

Rows of step: Clone row (1000 rows)

#	start_date	end_date	diff_dates	delta
1	2022-01-01	2027-12-31	2190	0
2	2022-01-01	2027-12-31	2190	1
3	2022-01-01	2027-12-31	2190	2
4	2022-01-01	2027-12-31	2190	3
5	2022-01-01	2027-12-31	2190	4
6	2022-01-01	2027-12-31	2190	5
7	2022-01-01	2027-12-31	2190	6
8	2022-01-01	2027-12-31	2190	7
9	2022-01-01	2027-12-31	2190	8
10	2022-01-01	2027-12-31	2190	9
11	2022-01-01	2027-12-31	2190	10
12	2022-01-01	2027-12-31	2190	11

- m. Add another Calculator step, and create a hop from the Clone row step to this one.
- n. Edit the new step, and add the field named a_single_date. As Calculation, select Date A + B Days. As Field A, select start_date and as Field B, select delta. Finally, as a Value type, select Date. For the rest of the columns, leave the default values.
- o. Run a final preview. You should see this:

Calculator

Step name: single date

Fields:

#	New field	Calculation	Field A	Field B	Field C	Val...	L...	P...	R...	Conversion ...
1	a_single_date	Date A + B Days	start_date	delta		Date			N	yyyy-MM-dd

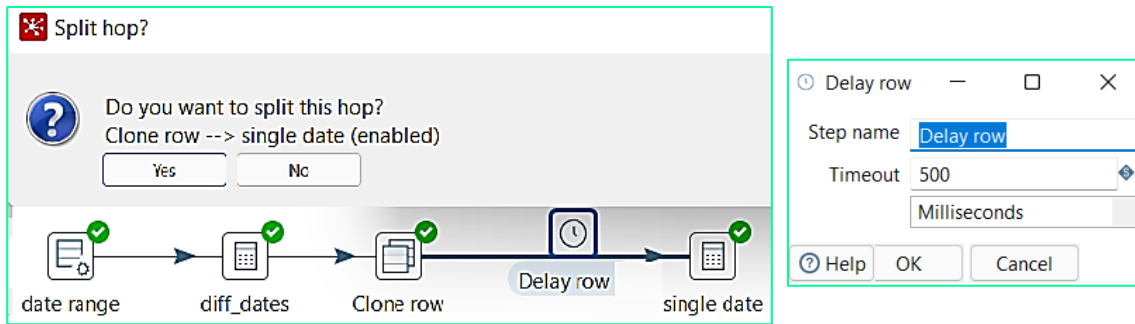
Execution Results

Logging Execution History Step Metrics Preview data

First rows Last rows Off

#	start_date	end_date	diff_dates	delta	a_single_date
1	2022-01-01	2027-12-31	2190	0	2022-01-01
2	2022-01-01	2027-12-31	2190	1	2022-01-02
3	2022-01-01	2027-12-31	2190	2	2022-01-03
4	2022-01-01	2027-12-31	2190	3	2022-01-04
5	2022-01-01	2027-12-31	2190	4	2022-01-05
6	2022-01-01	2027-12-31	2190	5	2022-01-06
7	2022-01-01	2027-12-31	2190	6	2022-01-07
8	2022-01-01	2027-12-31	2190	7	2022-01-08
9	2022-01-01	2027-12-31	2190	8	2022-01-09
1..	2022-01-01	2027-12-31	2190	9	2022-01-10
1..	2022-01-01	2027-12-31	2190	10	2022-01-11
1..	2022-01-01	2027-12-31	2190	11	2022-01-12
1..	2022-01-01	2027-12-31	2190	12	2022-01-13
1..	2022-01-01	2027-12-31	2190	13	2022-01-14
1..	2022-01-01	2027-12-31	2190	14	2022-01-15

- Now you will run the Transformation and inspect the data as the Transformation is being executed.
- Before doing that, we will do some changes to the Transformation so it runs slow, allowing us to see in detail what is happening:
 1. Edit the Generate Rows step and change the date range. As end_date, type 2023-12-31.
 2. From the Utility group of steps, drag to the work area the Delay row step. With this step, we will deliberately delay each row of data.
 3. Drag the step to the hop between the Clone row step and the second Calculator step, until the hop changes the width:
 4. A window will appear asking you if you want to split the hop. Click on Yes. The hop will be split in two: one from the Clone row step to the Delay row step, and the second one from this step to the Calculator step.
 5. Double-click on the Delay row step, and configure it using the following information: as Timeout, type 500, and in the drop-down list, select Milliseconds. Close the window.



6. Save the Transformation into "IS-545 lab M#4A NIM Debugging Date Range.ktr" and run it. You will see that it runs at a slower pace.

Now it is time to do the sniff testing, that is, looking at the rows that are coming into or out of a step in real time:

- Without stopping the execution, click on the second Calculator step. A popup window will show up describing the execution results of this step in real time. Ctrl-click two more steps: the Generate Rows step and the Clone row step. For each selected step, you will see the Step Metrics at runtime shown as output figure A.
- Now, let's inspect the data itself. Right-click on the second Calculator step and navigate to Sniff Test During Execution | Sniff test output rows. A window will appear showing the data as it's being generated.

In the Execution Results window, it's worth noting a column that we didn't mention before:

Column	Description
Speed (r/s)	The speed calculated in rows per second

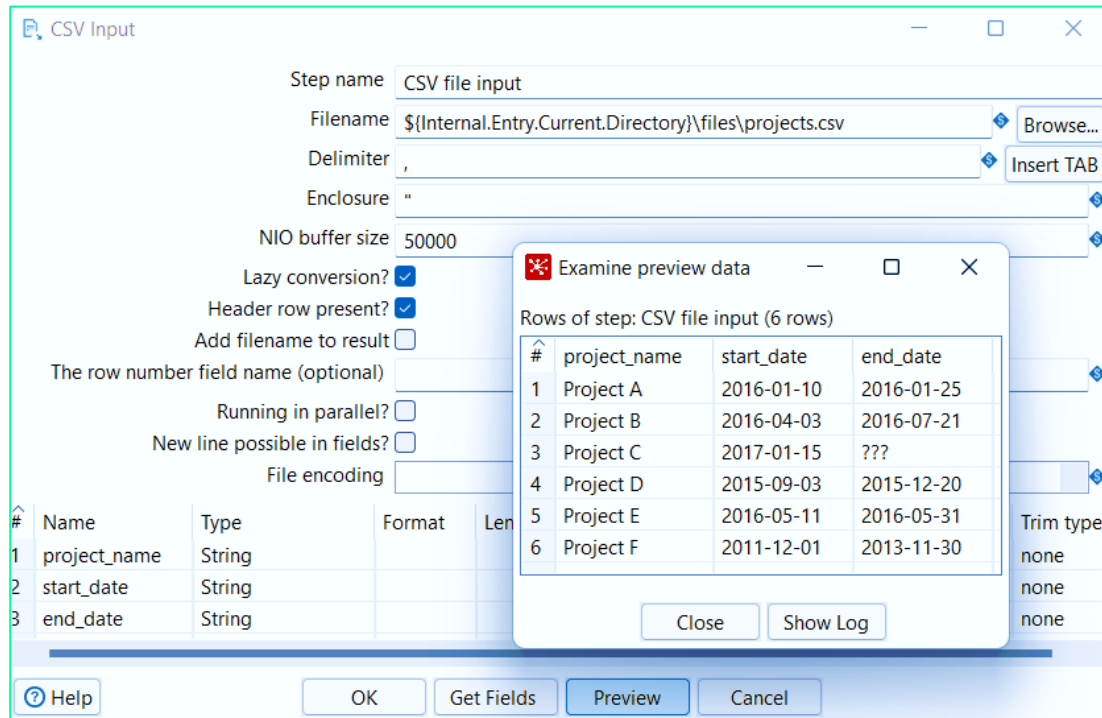
- As you put a delay of 500 milliseconds for each row, it's reasonable to see that the speed for the last step is two rows per second.

**** Note that sniff testing slows down the Transformation and its use is recommended just for debugging purposes.*

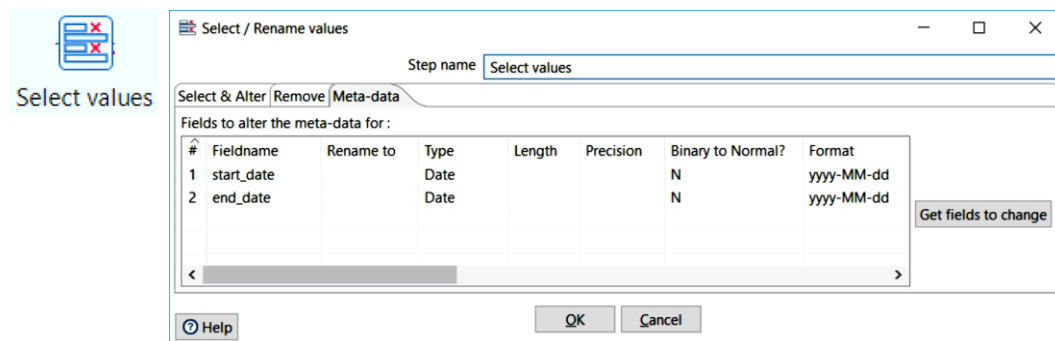
- While the Transformation was running, you experimented with the feature for sniffing the output rows. In the same way, you could have selected the Sniff test input rows option to see the incoming rows of data.

3. Implementing the Error Handling Functionality

- With the error handling functionality, you can capture errors that otherwise would cause the Transformation to halt. Instead of aborting, the rows that cause the errors are sent to a different stream for further treatment.
- The error handling functionality is implemented at step level. You don't need to implement error handling in every step. In fact, you cannot do that because not all steps support error handling.
- The objective of error handling is to implement it in the steps where it is more likely to have errors. In this case, we will work with the original version of the projects.txt file. *Remember that you removed an invalid row in that file.* Restore it and then proceed:
 - Open the Transformation of projects and save it under a different name. You can do it from the main menu by navigating to File | Save as... or from the main toolbar.
 - Edit the CSV file input step and change all the data types from Date to String. Also, delete the values under the Format column.



3. Now add a Select values step and insert it into the CSV file input step and the Calculator step.
4. Double-click on the Select values step and select the Meta-data tab. Fill the tab as follows:

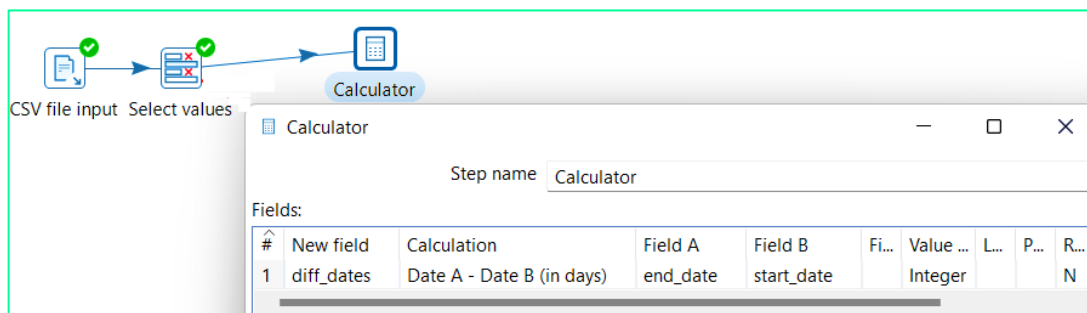


5. Close the window and run a preview. There is an error in the Select values step when trying to convert the invalid value ??? to a Date type:

`Select values.0 - end_date String<binary-string> : couldn't convert string [???`

Now let's get rid of that error by using the error handling feature:

1. Drag the Calculator step to the canvas:

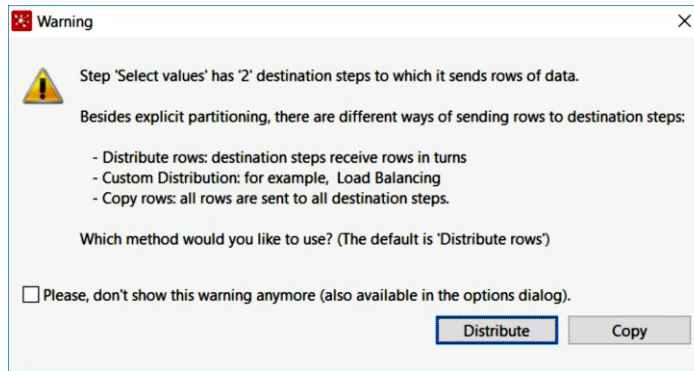


2. Drag to the canvas the Write to log step. You will find it in the Utility category of steps.

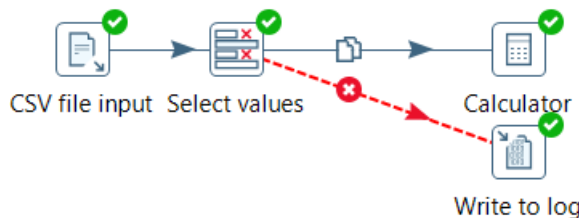
3. Create a new hop from the Select values step toward the Write to log step.
 - ♦ When asked for the kind of hop to create, select Error handling of step.



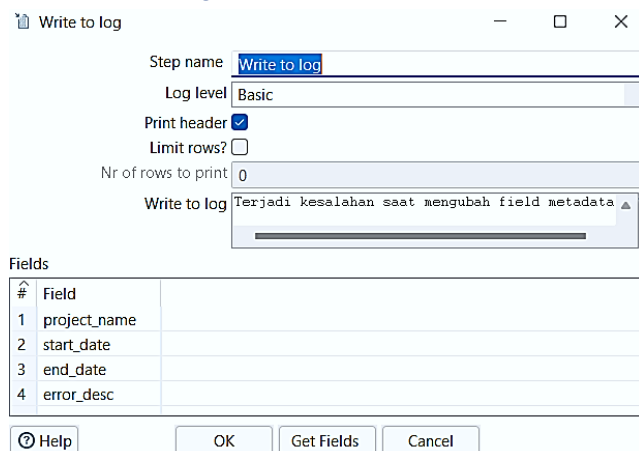
- ♦ Then, the following Warning window will appear:



4. Click on Copy.
5. Now your Transformation should look as shown in the following screenshot:



6. Double-click on the Write to log step. In the Write to log textbox, **Terjadi kesalahan saat mengubah field metadata**. Click on Get Fields. The grid will be populated with the names of the fields coming from the previous step.



Write to log

Step name: Write to log

Log level: Basic

Print header: ☒

Limit rows?: ☐

Nr of rows to print: 0

Write to log: Terjadi kesalahan saat mengubah field metadata

#	Field
1	project_name
2	start_date
3	end_date
4	error_desc

Help OK Get Fields Cancel

7. Add more two steps :
 - ♦ Number Ranges, to create performance range of lower and upper bound of different date
 - ♦ User Defined Java Expression to give an messages to the performance range

Number ranges

Step name: Performance Range

Input field: diff_dates

Output field: performance

Default value(if no range): unknown

Ranges (min <= x < max):

#	Lower Bound	Upper Bound	Value
1		30.0	excellent
2	30.0	80.0	very good
3	80.0	160.0	good
4	160.0		poor

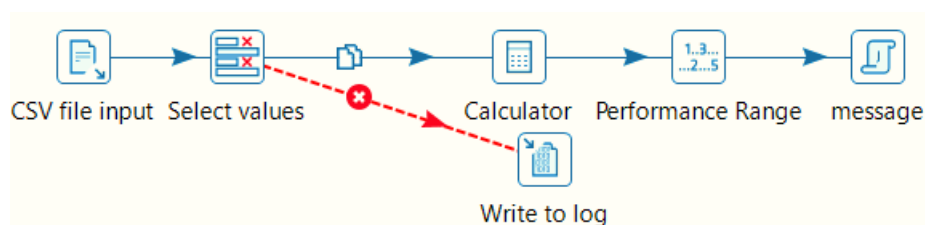
User Defined Java Expression

Step name: message

Fields: message

#	New field	Java expression	Value type
1	duration	(diff_dates == null)?"unknown":diff_dates + " days"	String
2	message	"The performance was " + performance	String

8. Close the window and save the Transformation as "IS-545 Lab M#4B NIM Error Handling.ktr". Now run it. Look at the Logging tab in the Execution Results window. The log will look like this:



9. Run the transformation program and now, preview the data in each step with the output according to the output figure B
10. You can see in the Calculator step, all the lines except the line containing the invalid date. This output is exactly the same as the one in the screenshot Preview of a Transformation.
11. On the Write to log step. You will only see the line that had the invalid end_date value for project C and only.
- ❖ Preview of data with errors With just a couple of clicks, you redirected the rows with errors to an alternative stream, represented by the hop in red. As you could see, both in the preview and in the Execution Results windows, the rows with valid values continued their way towards the Calculator step, while the row whose end_date field could not be converted to Date went to the Write to log step. In the Write to log step, you wrote an informative message as well as the values for all the fields, so it was easy to identify which row (or rows) caused this situation.

4. Getting data from relational databases

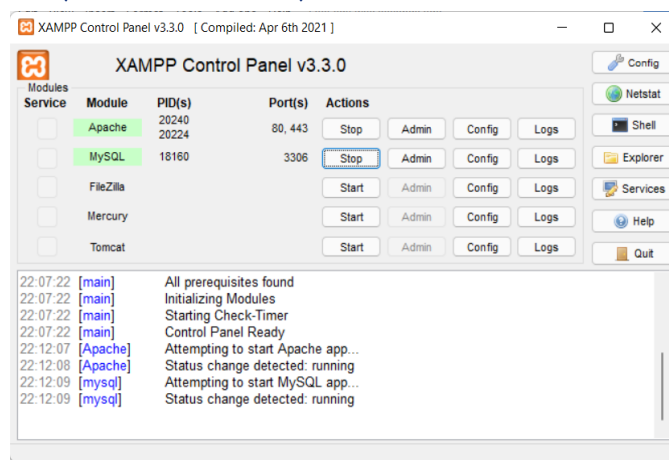
- ▶ PDI has the ability to connect to both commercial RDBMS such as Oracle or MS SQL Server, and free RDBMS such as MySQL or PostgreSQL. In order to interact with a particular database, you have to define a connection to it.
- ▶ However, before you make connections and use the database in the job and transformation processes, you must prepare the DBMS first.
- ▶ One of the open source DBMS products that you have known and used so far is XAMPP but here below is a guide to installing and using XAMPP for those of you who have never used it.

XAMPP Installation

- ▶ First, it can be downloaded via the official website at the following link, <http://www.apachefriends.org/en/index.html>.
 - ▶ After the file is successfully downloaded, the next step is to extract the file and start the installation stage according to the instructions and instructions provided. Can choose the required components according to the needs of the software to be developed.
 - ▶ Next, you must determine your Xampp storage directory file by paying attention to the program's storage capacity. We recommend putting it on your local disk system for easier maintenance.
 - ▶ Next, you can wait for the installation process to finish and Xampp is ready to run offline using the help of localhost.
1. Open XAMPP Control Panel. If you don't have a Desktop or Quick Launch icon, Click Start > All Programs > XAMPP > XAMPP Control Panel.



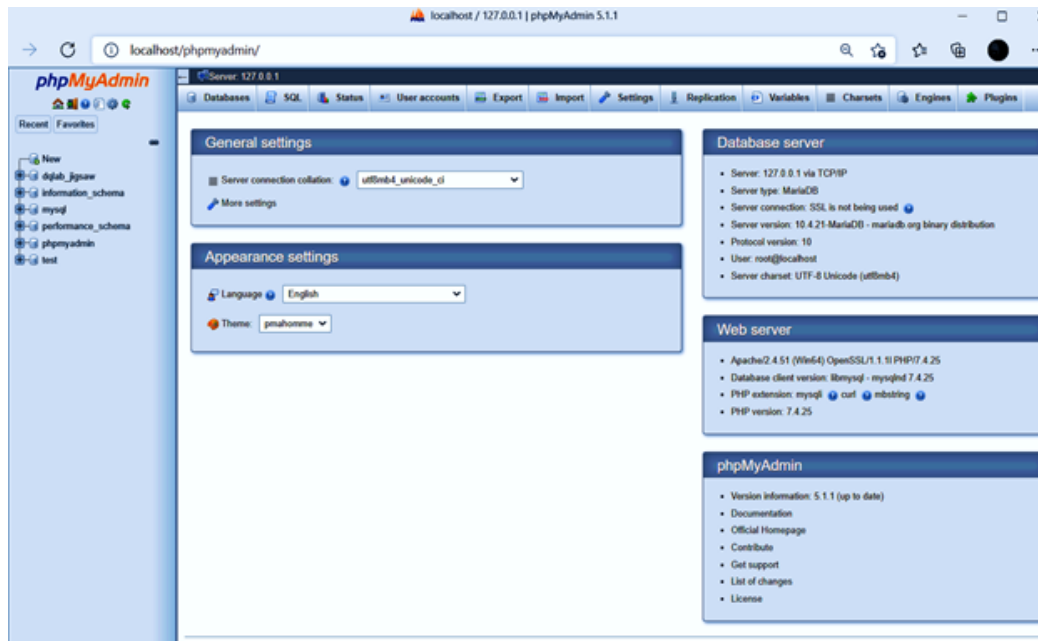
2. For PDI to be able to connect to the MySQL database, click the button that says start to run the Apache and MySQL modules (for database needs).



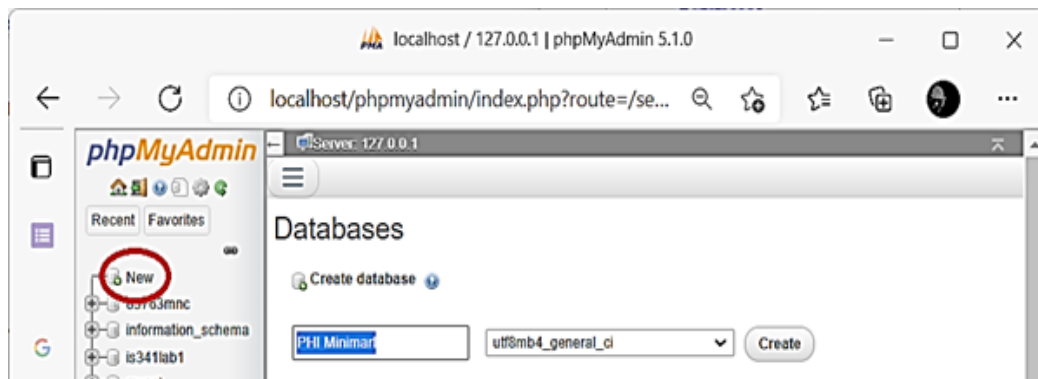
- After you have confirmed that the application can run without any error messages, then you can access localhost by writing the URL address in your web browser or via PDI which is will be next discussion.

Database Creation and Data Import in XAMPP

- The following are some steps to prepare a database and its data in the form of phi_minimart sales transaction data for later exploration through PDI.
- Connection to Database environment, click the MySQL Admin button to enter MySQL database settings via browser:

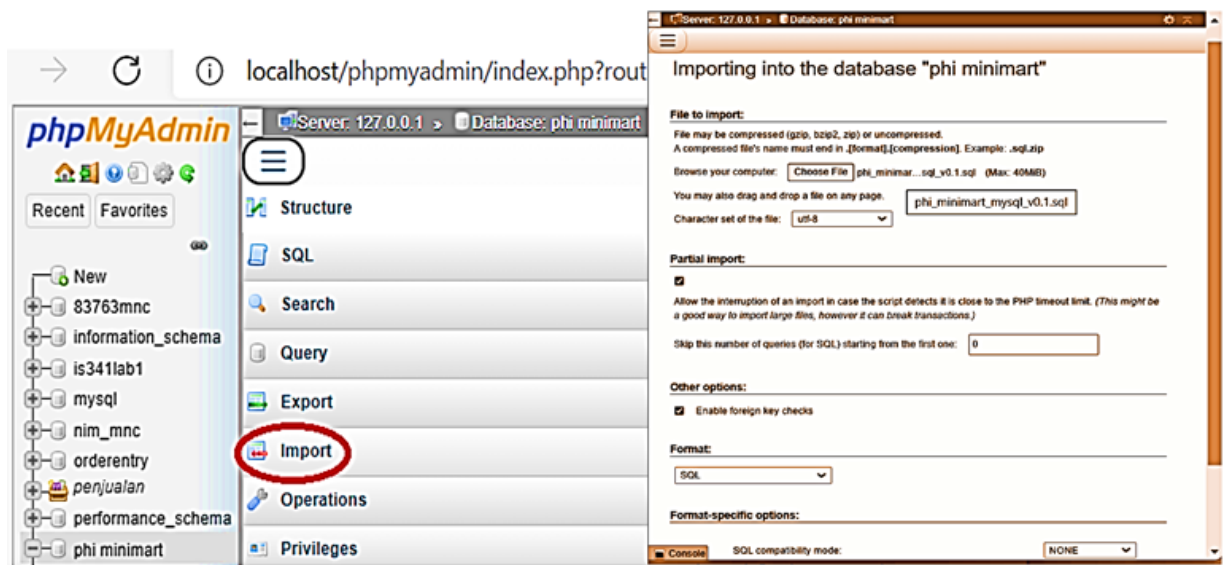


- Create PHI Minimart Database, click New → name it PHI Minimart → click create:

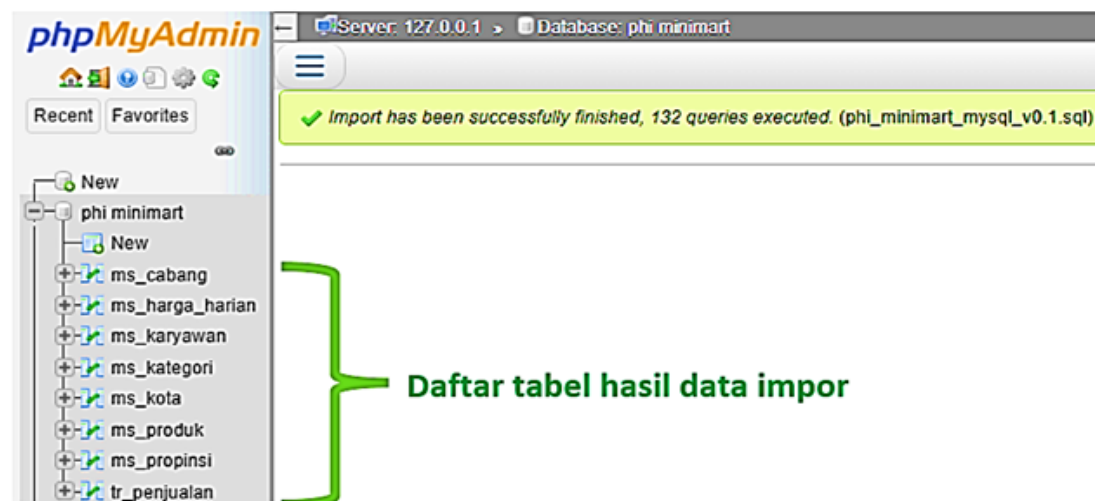


After clicking create, the PHI Minimart database is listed in the database list to the right of the XAMPP workbench.

- Import PHI Minimart data through these steps: click the row button at the top, then select import to import PHI Minimart data from the `phi_minimart_mysql_v0.1.sql` file and then click go.



4. If the data import process is successful, then the PHI Minimart Database now contains several tables as can be seen in the following screenshot:



5. It is also possible to create and import the above data using the SQuLYog application which has been installed in module-2 "Introduction to the Pentaho Data Integration Application: Job and Transformation", SQuLYog is a GUI/Graphical User Interface for database management but does not have a MySQL database engine.
6. After the PHI Minimart database has been successfully built, complete with the data contents of each table, the next step is to connect via PDI.

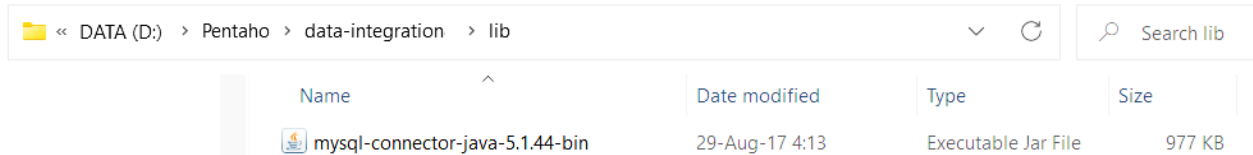
Connect to MySQL database and use database explorer

There are two things you must do to connect to the database, if you want to use its data inside PDI:

- ▶ Install proper JDBC driver
- ▶ Establish connection to database

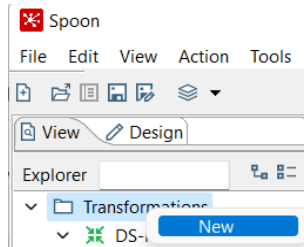
1. Make sure to have JDBC driver, .jar file – for PDI which can be downloaded via url: [mysql-connector-java JAR 5.1.44](https://dev.mysql.com/downloads/connector-java/) with all dependencies!

- Once you have it, you must copy it to the /lib folder in the PDI installation directory and restart the Spoon to activate the JDBC driver.

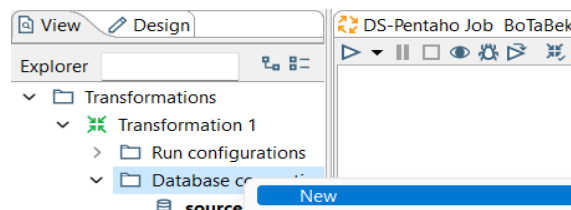


The above JDBC driver installation only needs to be done once in a PDI environment. Once the driver is installed, you can establish a connection to the database, as follows:

- Create a new transformation.

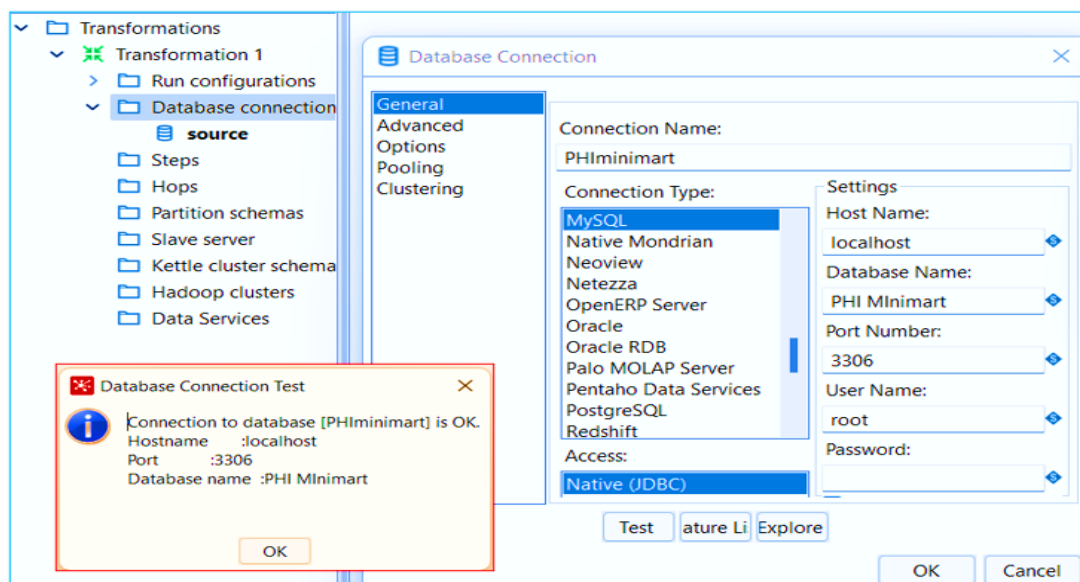


- Click on the View tab, right-click on the Database connection option, and select New. A Database Connection window will appear.

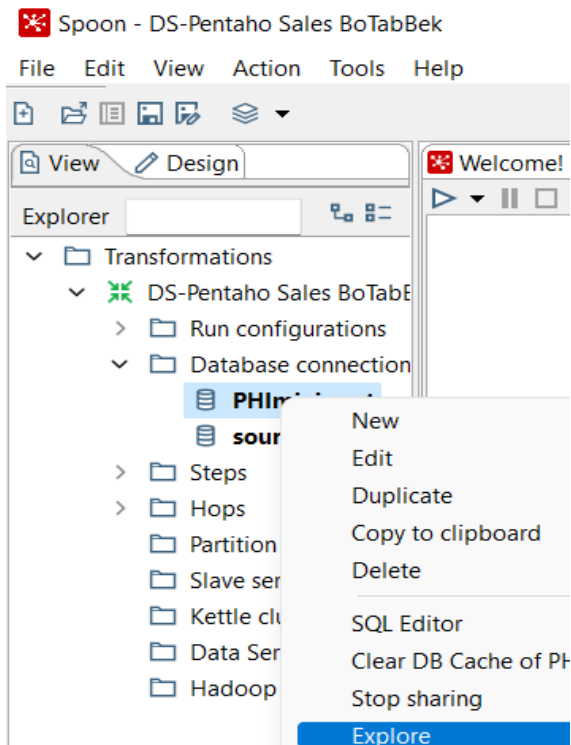


Under Connection type:, select the machine that matches yours. Fill in the text boxes in the Settings area with the appropriate values: Host Name, Database Name, Port Number, User Name, and Password.

- See the following database connection, configured for the phi minimart database, hosted on the MySQL database on the local machine.
- Click Test to verify that the connection has been established properly, and that it can reach the database from PDI.

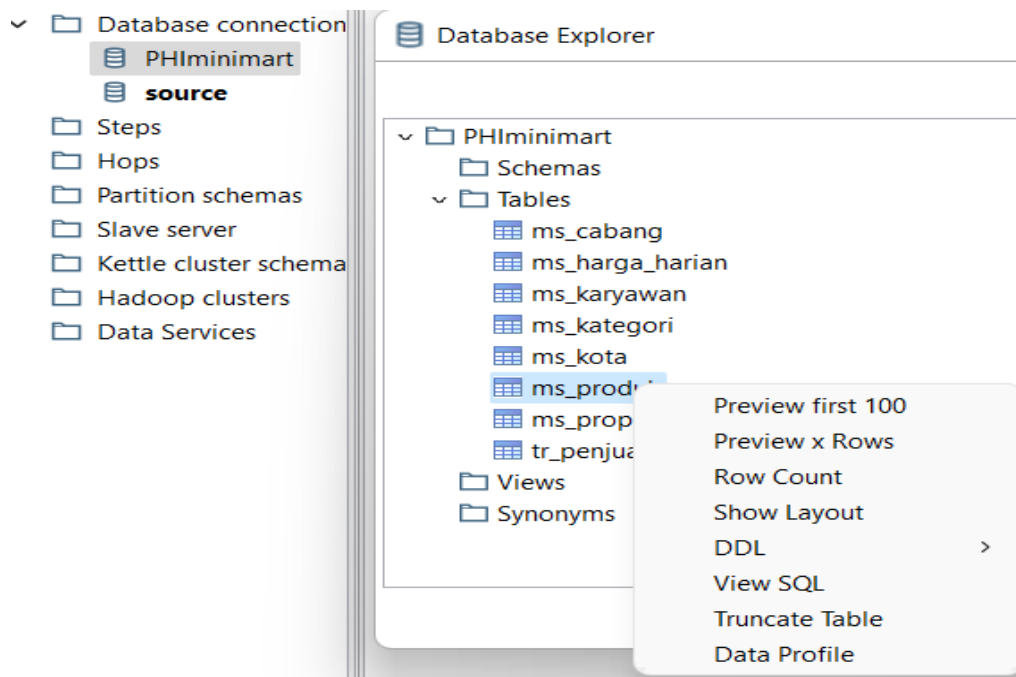


5. Click Explorer as shown in the image below, its location is on the Views tab which is the explorer function for Transformations, continue clicking "database connection" to go to the PHMinimart database that has been created.

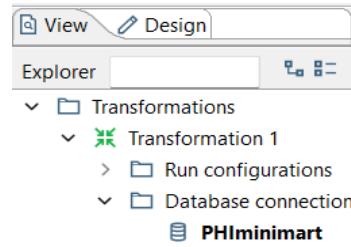


Database Explorer allows to explore tables and views in the database, and to run queries and DDL statements.

6. Right click on any table name. A pop-up menu will appear, offering several options, as shown in the following screenshot:



7. If you choose Preview first 100, PDI will display the contents of data per 100 records as shown the output figure C. Screenshotted and save it into "IS-545 Lab M\$4C NIM PHIminimart.jpg", then close the Database Explorer. Click OK.
 - ▶ A new database connection will appear under the database connection item.
 - ▶ The database connection you just defined will be available in the current transformation, as we will see soon. If you want it to be available in a new transformation, you must share it, as follows:
 - ♦ Right click on the database connection and click Share.
8. Connection name will be bolded, meaning definition will be available outside current transformation scope:



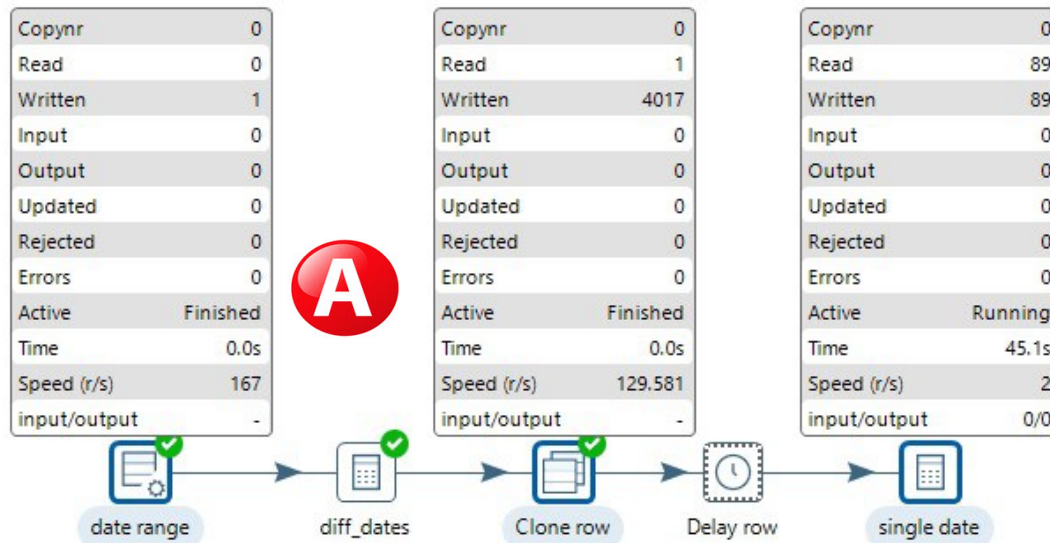
- ➡ At the end of today's practicum, collect all your files:
 - ♦ IS-545 lab M#4A NIM Debugging Date Range.ktr
 - ♦ IS-545 Lab M#4B NIM Error Handling.ktr
 - ♦ IS-545 Lab M\$4C NIM PHIminimart.jpg
 into a zip file with name convention "IS-545 Lab M#4 NIM yourName.zip" and submit to the e-Learning IS-545 Practicum M#4.

HASIL/ LUARAN

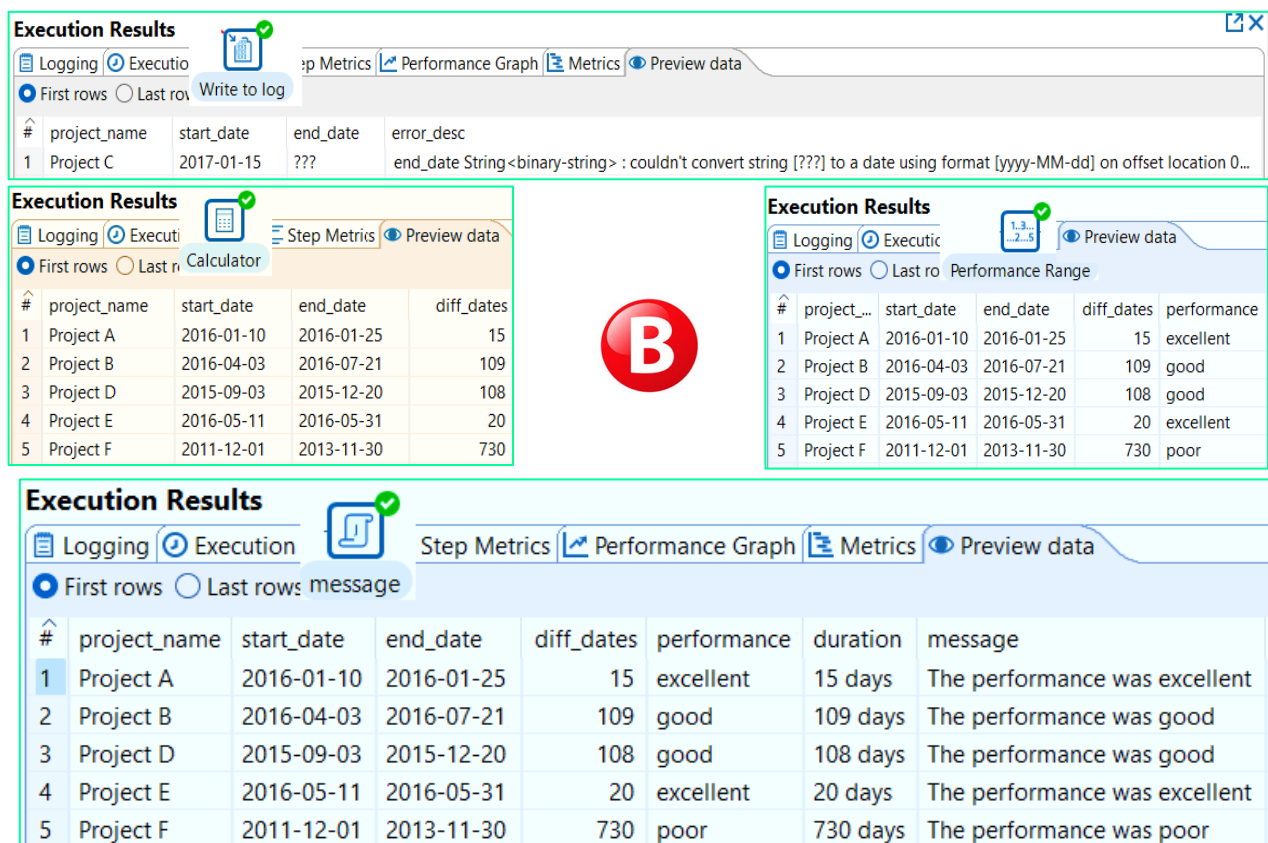
1. PDI Logging setup

No submission

2. Debugging Tips for Transformations and Jobs



3. Implementing the Error Handling Functionality



Execution Results - View 1

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

First rows | Last rows | Write to log

#	project_name	start_date	end_date	error_desc
1	Project C	2017-01-15	???	end_date String<binary-string> : couldn't convert string [??] to a date using format [yyyy-MM-dd] on offset location 0...

Execution Results - View 2

Logging | Execution | Step Metrics | Preview data

First rows | Last rows | Calculator

#	project_name	start_date	end_date	diff_dates
1	Project A	2016-01-10	2016-01-25	15
2	Project B	2016-04-03	2016-07-21	109
3	Project D	2015-09-03	2015-12-20	108
4	Project E	2016-05-11	2016-05-31	20
5	Project F	2011-12-01	2013-11-30	730

Execution Results - View 3

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

First rows | Last rows | message

#	project_name	start_date	end_date	diff_dates	performance	duration	message
1	Project A	2016-01-10	2016-01-25	15	excellent	15 days	The performance was excellent
2	Project B	2016-04-03	2016-07-21	109	good	109 days	The performance was good
3	Project D	2015-09-03	2015-12-20	108	good	108 days	The performance was good
4	Project E	2016-05-11	2016-05-31	20	excellent	20 days	The performance was excellent
5	Project F	2011-12-01	2013-11-30	730	poor	730 days	The performance was poor

4. Getting data from relational databases

PHMinimart

- Schemas
- Tables
 - ms_cabang
 - ms_harga_harian
 - ms_karyawan
 - ms_kategori
 - ms_kota
 - ms_produk
 - ms_propinsi
 - tr_penjualan
 - Views
 - Synonyms

Examine preview data

Rows of step: ms_harga_harian (100 rows)

#	kode_produk	tgl_berlaku	kode_cabang	harga_berlaku...	modal_cabang	biaya_cabang
1	PROD-0000001	2008/01/01 00:00:00.000000000	CABANG-039	10980	10380	190
2	PROD-0000002	2008/01/01 00:00:00.000000000	CABANG-039	4220	4040	10
3	PROD-0000003	2008/01/01 00:00:00.000000000	CABANG-039	6220	5900	90
4	PROD-0000004	2008/01/01 00:00:00.000000000	CABANG-039	8020	7620	60
5	PROD-0000005	2008/01/01 00:00:00.000000000	CABANG-039	7210	6860	10
6	PROD-0000006	2008/01/01 00:00:00.000000000	CABANG-039	8110	7770	10
7	PROD-0000007	2008/01/01 00:00:00.000000000	CABANG-039	4640	4440	20
8	PROD-0000008	2008/01/01 00:00:00.000000000	CABANG-039	1060	1000	0
9	PROD-0000009	2008/01/01 00:00:00.000000000	CABANG-039	4690	4500	80
1..	PROD-0000010	2008/01/01 00:00:00.000000000	CABANG-039	1100	1040	30
1..	PROD-0000011	2008/01/01 00:00:00.000000000	CABANG-039	1210	1150	30
1..	PROD-0000012	2008/01/01 00:00:00.000000000	CABANG-039	3440	3250	30
1..	PROD-0000013	2008/01/01 00:00:00.000000000	CABANG-039	12280	11620	120
1..	PROD-0000014	2008/01/01 00:00:00.000000000	CABANG-039	11520	11060	60
1..	PROD-0000015	2008/01/01 00:00:00.000000000	CABANG-039	18880	17940	10

Close

The End

REFERENSI

-Utama-

1. María Carina Roldán. 2017. Learning Pentaho Data Integration 8 CE Third Edition. Packt Publishing.
2. Matt Casters, Roland Bouman, Jos van Dongen. 2010. Pentaho® Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration Wiley Publishing, Inc.
3. Pentaho Community website [Data Integration \(hitachivantara.com\)](http://hitachivantara.com)

-Pendukung-

4. Other additional information/knowledge that can be obtained from certain urls/websites.