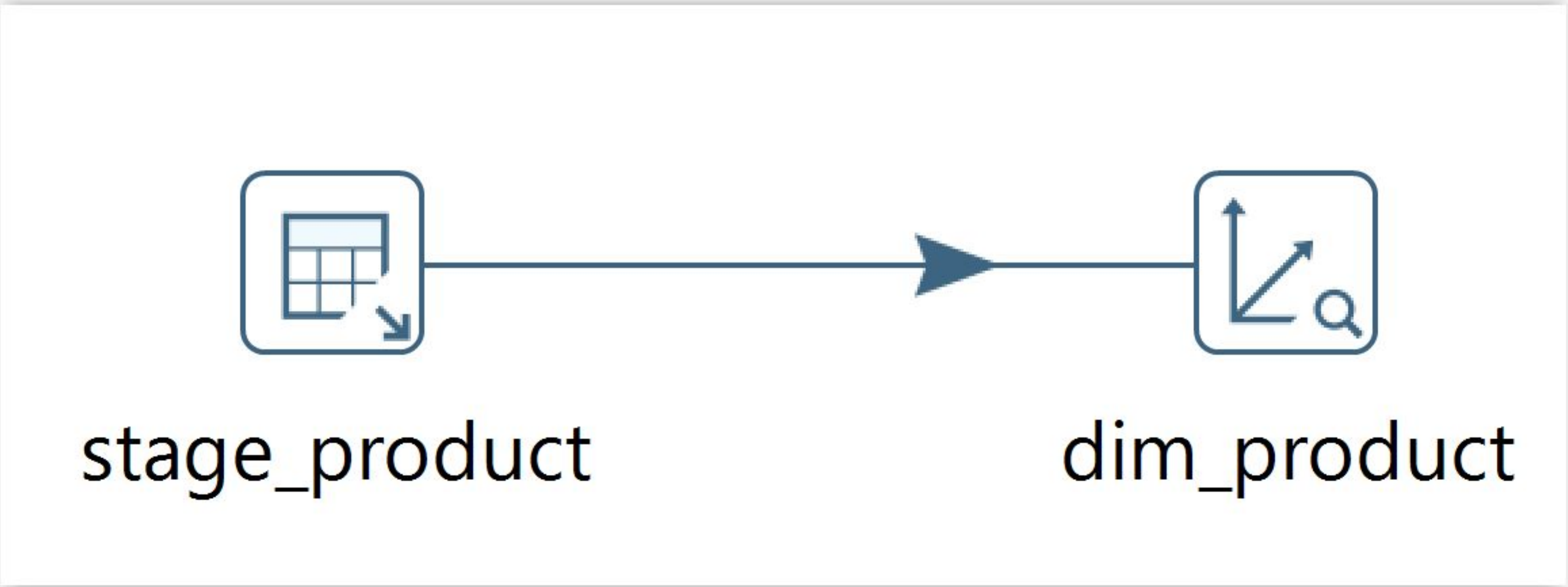
Building Sales Data mart Using Pentaho Part 2 (continued)

By Naheed Anjum Arafat

Task 1.2

`Stage_product → dim_product`

Objective



Task 1.2 - Create Transformation

- Save as `dim_product.ktr` in folder `dim_product`

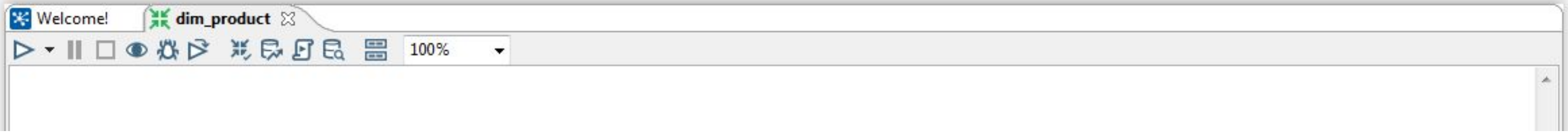
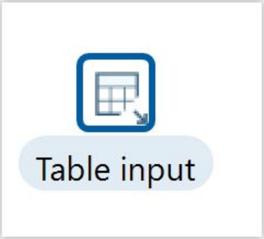
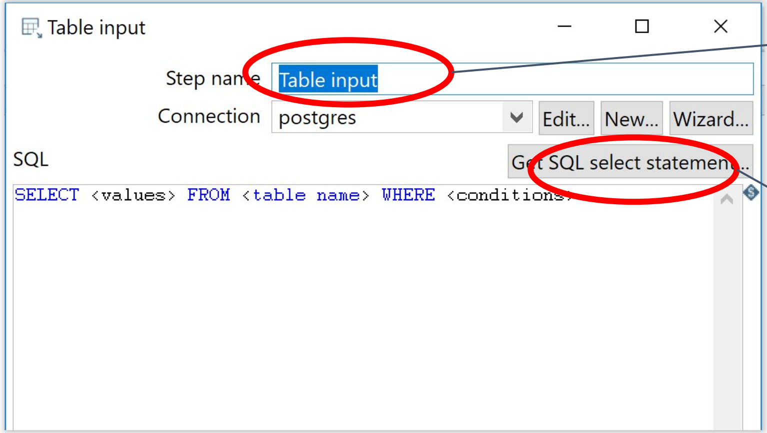


Table input





Step name: Table input

Connection: postgres

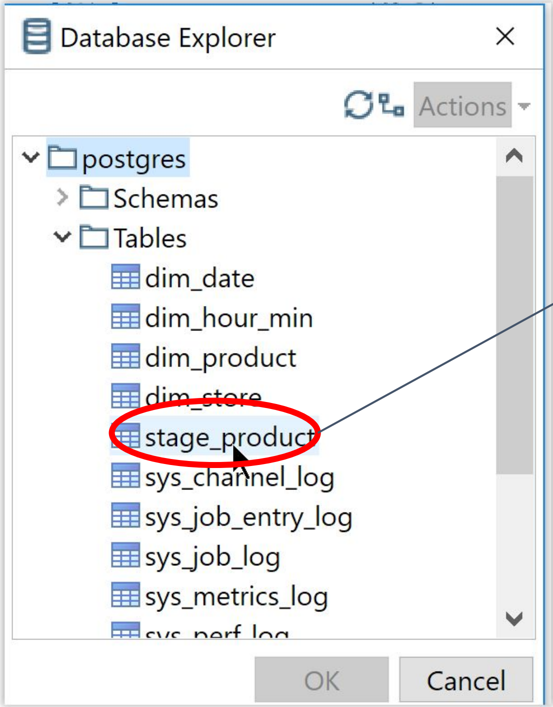
SQL

```
SELECT <values> FROM <table name> WHERE <conditions>
```

Set Step name to `stage_product`

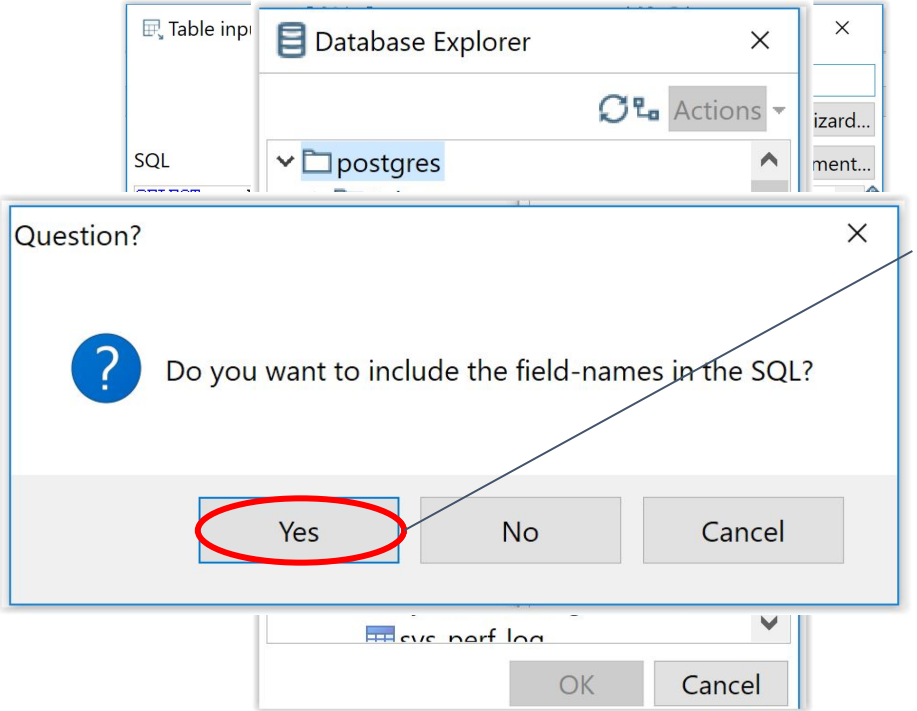
Click on Get SQL select statement

Table input



Double-click on `stage_product`

Table input

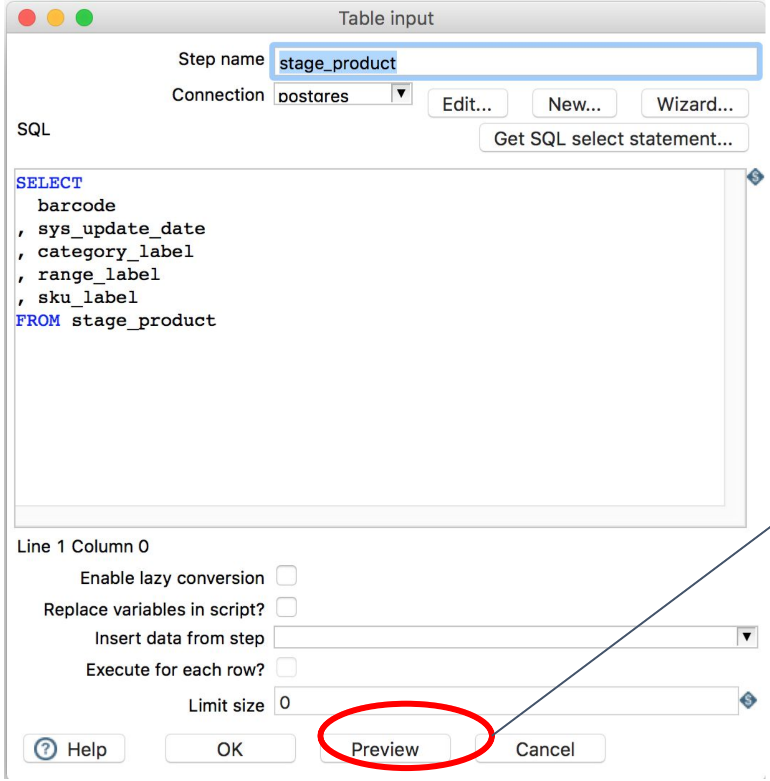


Question?

Do you want to include the field-names in the SQL?

Click on YES

Table input



Step name: stage_product

Connection: postgres

```
SELECT
  barcode
, sys_update_date
, category_label
, range_label
, sku_label
FROM stage_product
```

Line 1 Column 0

Enable lazy conversion □

Replace variables in script? □

Insert data from step:

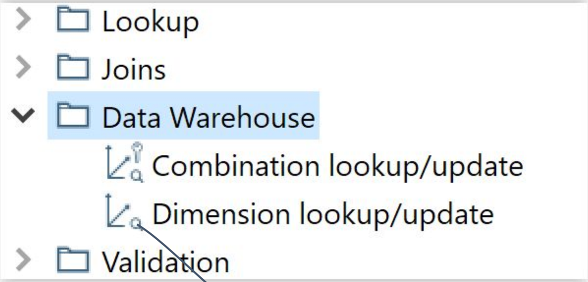
Execute for each row? □

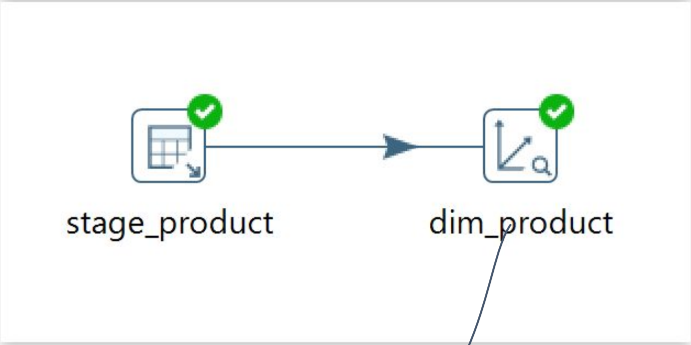
Limit size: 0

Click on Preview to see the contents of the table

Dimension lookup step

- Drag dimension lookup step to the canvas
- Connect from sys_update_date to lookup step

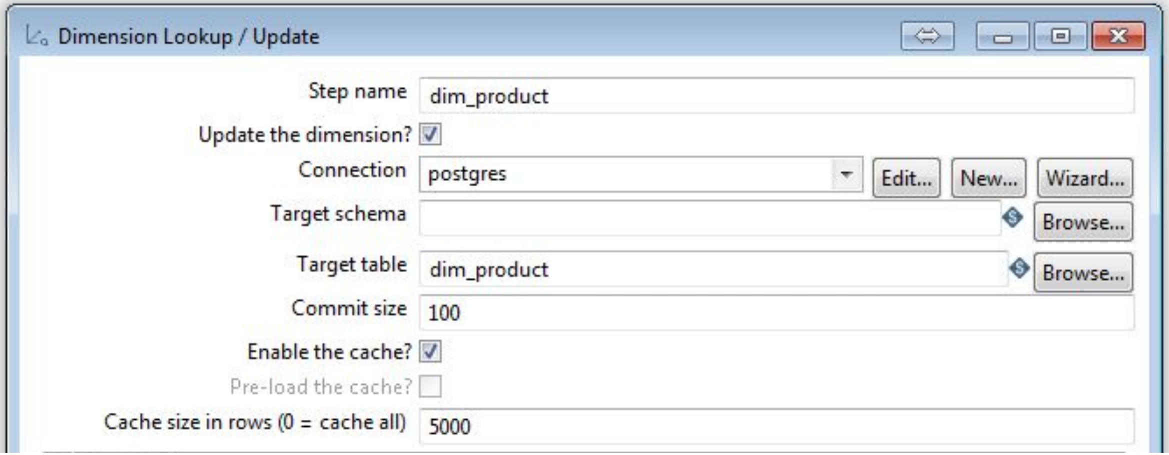




Drag and drop
Dimension lookup/update step

Edit Dimension Lookup

- Write “Step name” : `dim_product`
- Choose “connection” : `postgres`
- Write “target table” : `dim_product`



Step name: dim_product

Update the dimension? ☑

Connection: postgres

Target schema:

Target table: dim_product

Commit size: 100

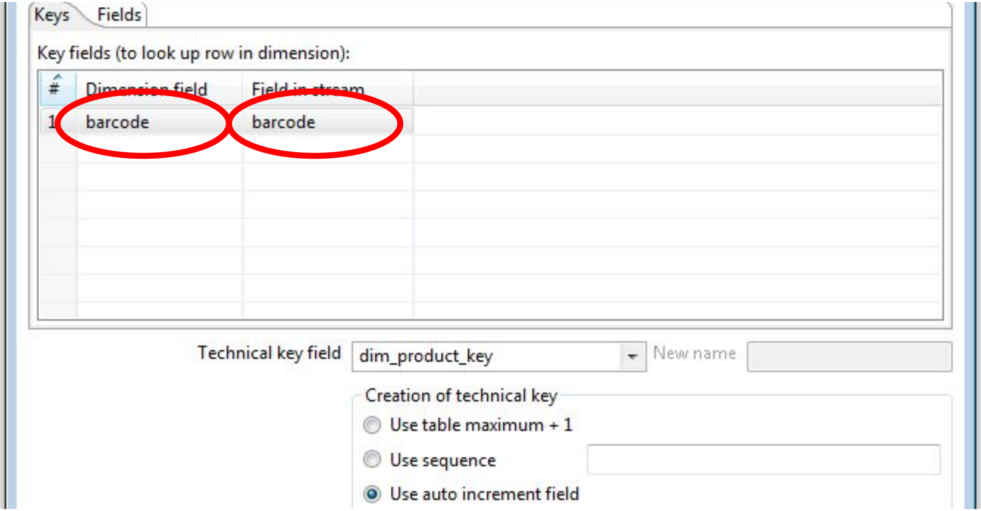
Enable the cache? ☑

Pre-load the cache? □

Cache size in rows (0 = cache all): 5000

Edit Dimension Lookup

- In Keys Tab, Click on field in stream and choose “barcode”
- Write “barcode” to the dimension field



Key fields (to look up row in dimension):

#	Dimension field	Field in stream
1	barcode	barcode

Technical key field: dim_product_key

New name:

Creation of technical key

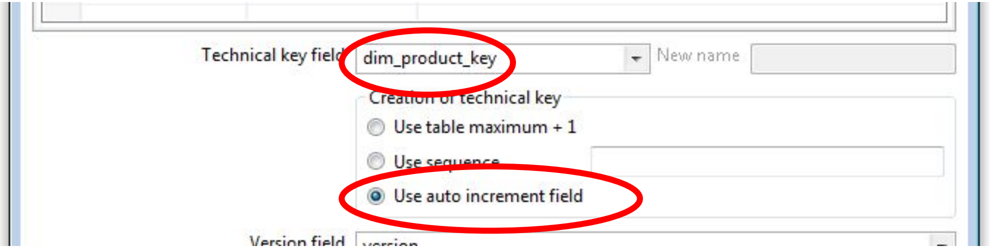
○ Use table maximum + 1

○ Use sequence

● Use auto increment field

Edit Dimension Lookup

- Set technical key field to `dim_product_key`
- Check out `Use auto increment field`



Technical key field dim_product_key

Creation of technical key

○ Use table maximum + 1

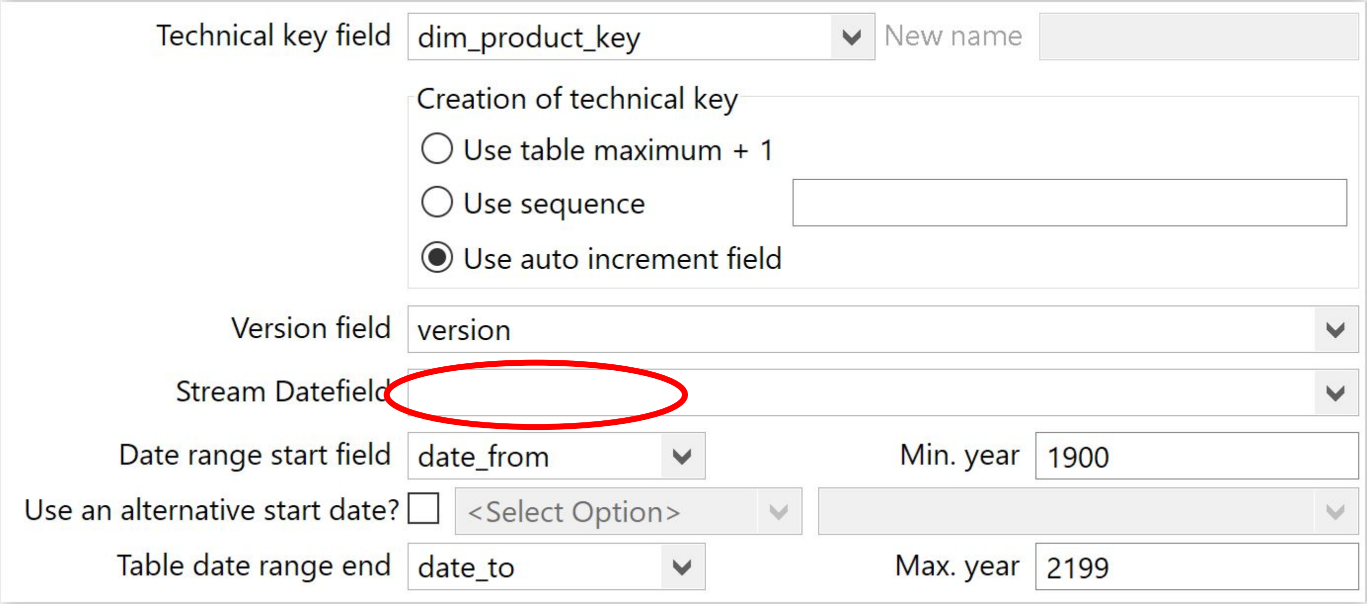
○ Use sequence

● Use auto increment field

Version field version

Edit Dimension Lookup

- Make sure `Stream datefield` is empty



Technical key field: dim_product_key

New name:

Creation of technical key

○ Use table maximum + 1

○ Use sequence

● Use auto increment field

Version field: version

Stream Datefield:

Date range start field: date_from

Min. year: 1900

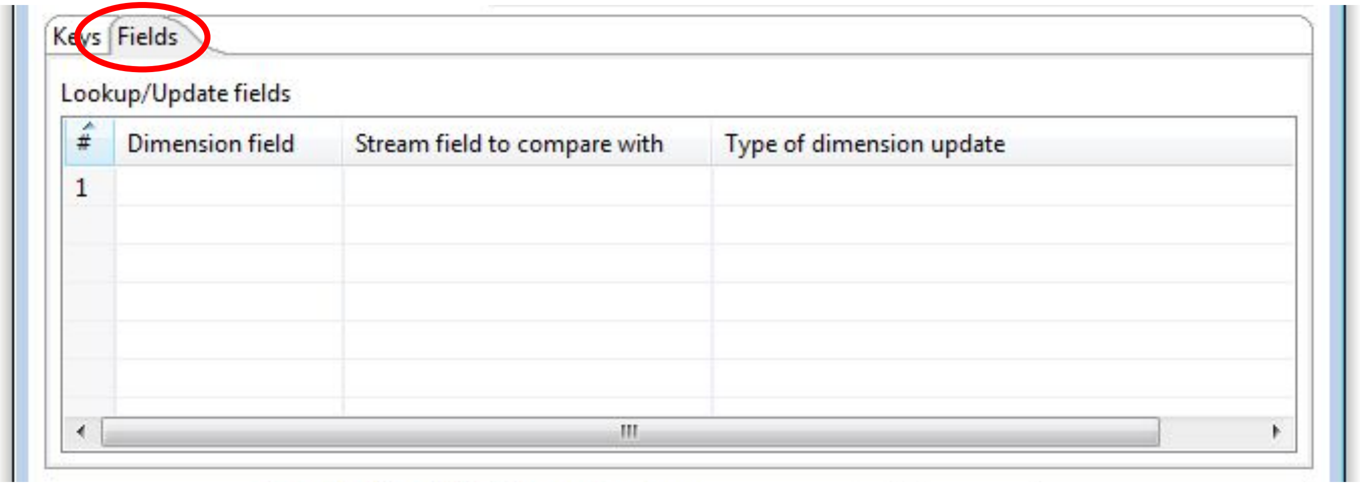
Use an alternative start date? □ <Select Option>

Table date range end: date_to

Max. year: 2199

Dimension attribute fields

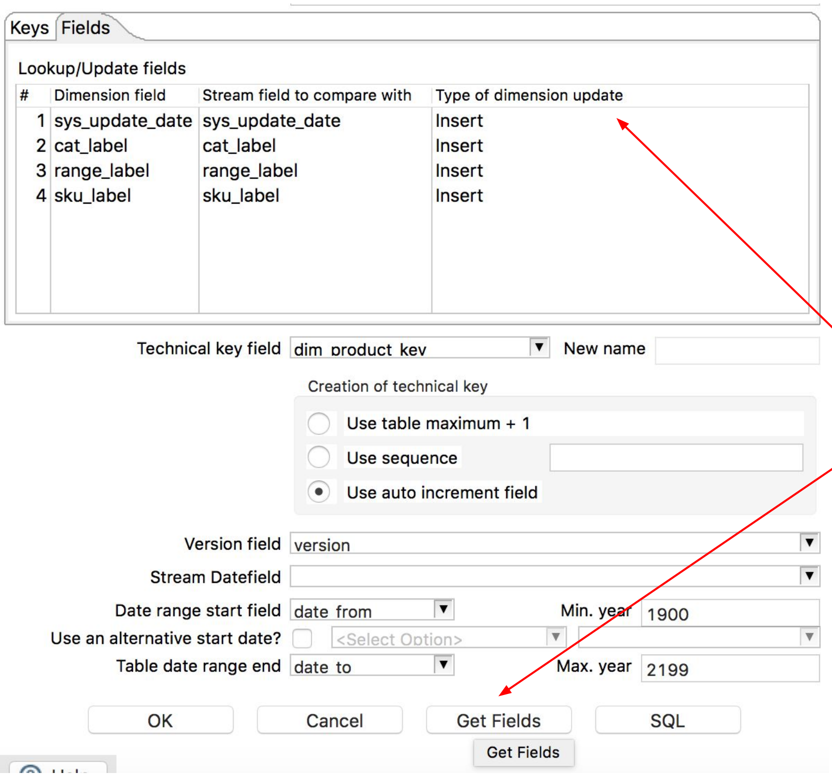
- Click on `Fields` tab to add dimension attributes



Lookup/Update fields

#	Dimension field	Stream field to compare with	Type of dimension update
1			

Dimension attribute fields



Lookup/Update fields

#	Dimension field	Stream field to compare with	Type of dimension update
1	sys_update_date	sys_update_date	Insert
2	cat_label	cat_label	Insert
3	range_label	range_label	Insert
4	sku_label	sku_label	Insert

Technical key field: dim_product_key New name:

Creation of technical key

○ Use table maximum + 1

○ Use sequence

● Use auto increment field

Version field: version

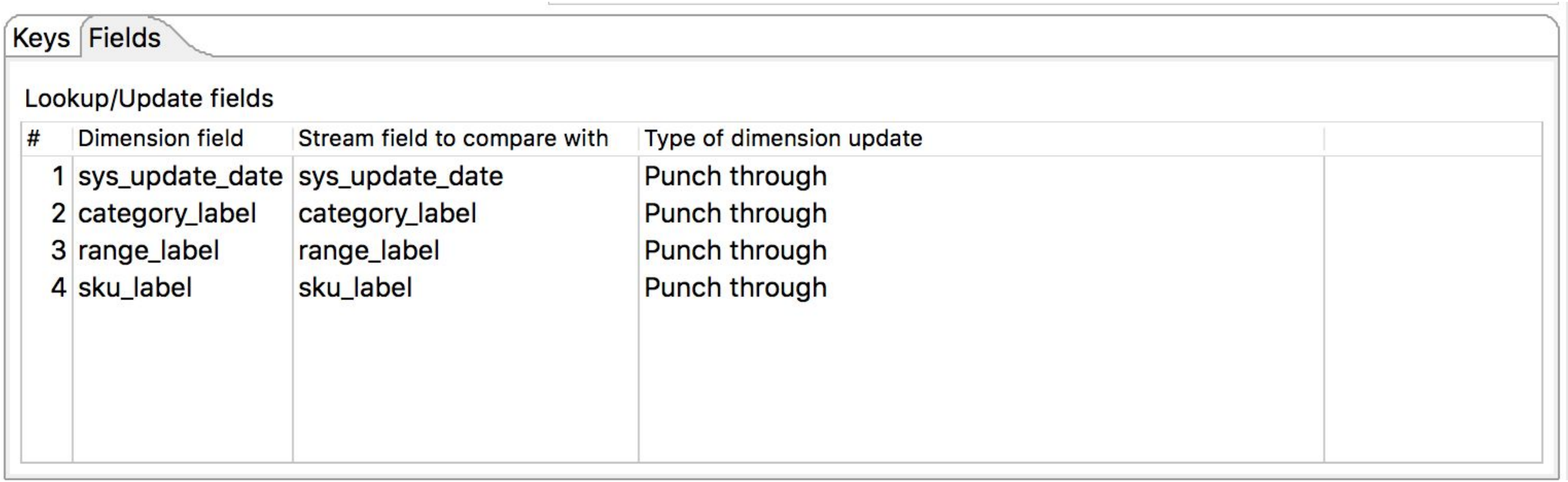
Stream Datefield:

Date range start field: date_from Min. year: 1900

Use an alternative start date? □ <Select Option>

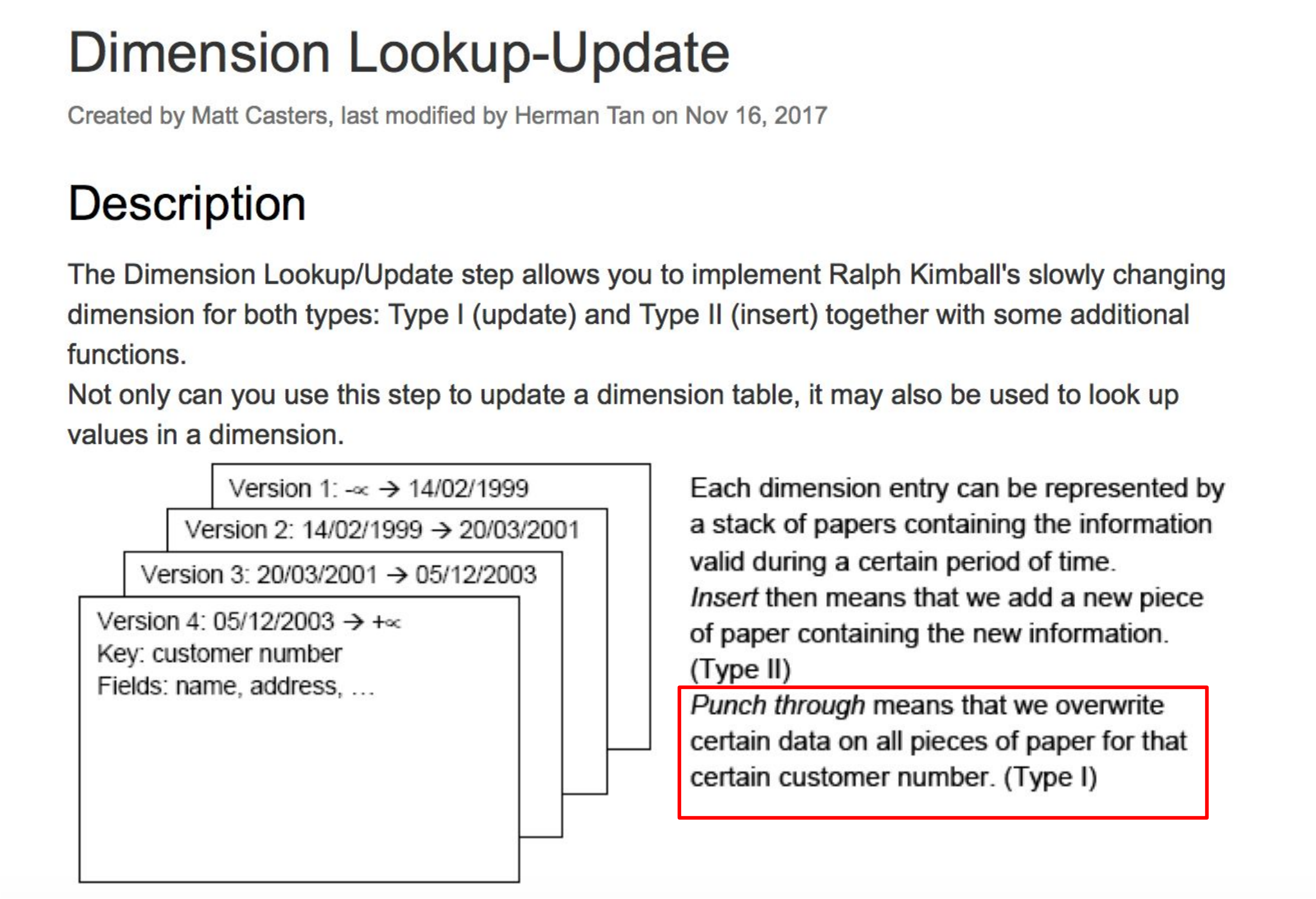
Table date range end: date_to Max. year: 2199

- Click Get Fields
- Change Type of dimension update “`Punch through`” for Kimbals Type 1 update



Lookup/Update fields

#	Dimension field	Stream field to compare with	Type of dimension update
1	sys_update_date	sys_update_date	Punch through
2	category_label	category_label	Punch through
3	range_label	range_label	Punch through
4	sku_label	sku_label	Punch through



Dimension Lookup-Update

Created by Matt Casters, last modified by Herman Tan on Nov 16, 2017

Description

The Dimension Lookup/Update step allows you to implement Ralph Kimball's slowly changing dimension for both types: Type I (update) and Type II (insert) together with some additional functions.
Not only can you use this step to update a dimension table, it may also be used to look up values in a dimension.

Version 1: -∞ → 14/02/1999

Version 2: 14/02/1999 → 20/03/2001

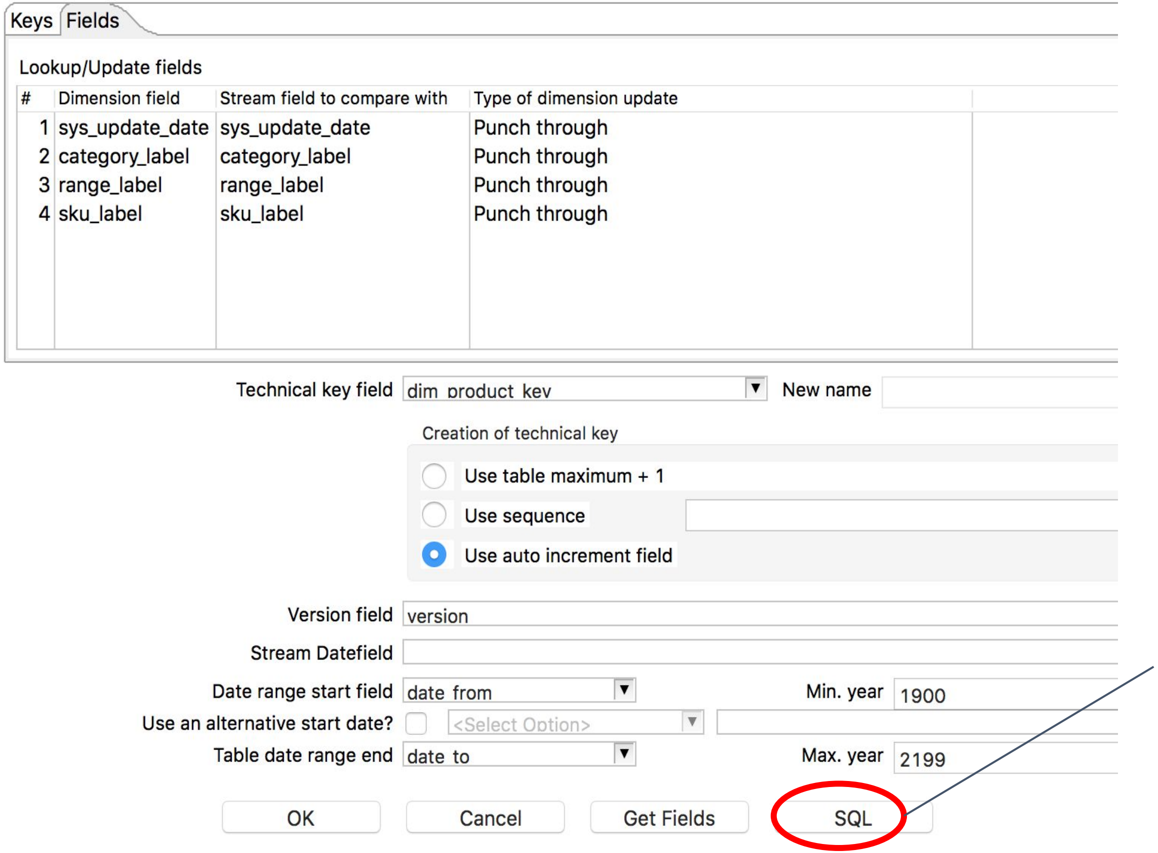
Version 3: 20/03/2001 → 05/12/2003

Version 4: 05/12/2003 → +∞
Key: customer number
Fields: name, address, …

Each dimension entry can be represented by a stack of papers containing the information valid during a certain period of time.
Insert then means that we add a new piece of paper containing the new information. (Type II)

Punch through means that we overwrite certain data on all pieces of paper for that certain customer number. (Type I)

Create Dimension Table



Lookup/Update fields

#	Dimension field	Stream field to compare with	Type of dimension update
1	sys_update_date	sys_update_date	Punch through
2	category_label	category_label	Punch through
3	range_label	range_label	Punch through
4	sku_label	sku_label	Punch through

Technical key field: dim_product_key
New name:

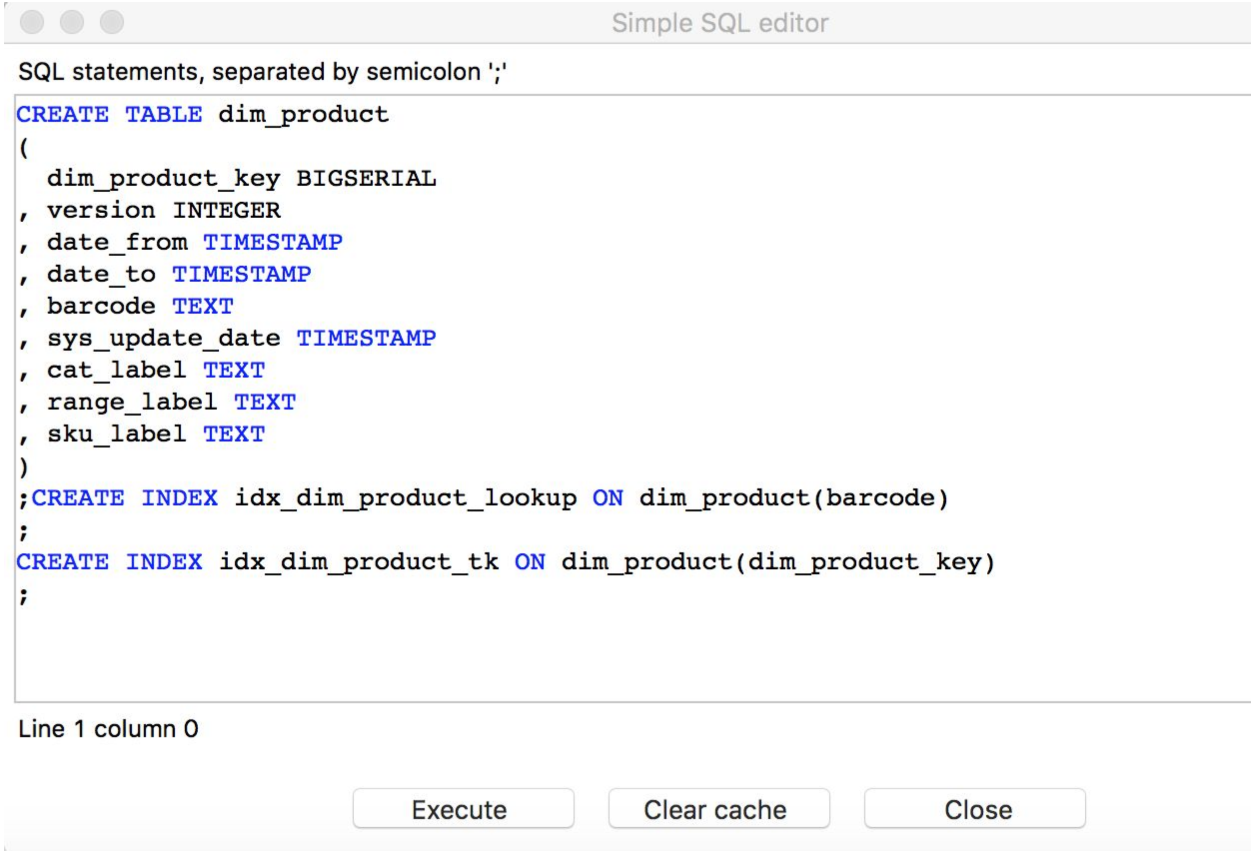
Creation of technical key

○ Use table maximum + 1
○ Use sequence
● Use auto increment field

Version field: version
Stream Datefield:
Date range start field: date_from
Min. year: 1900
Use an alternative start date? □ <Select Option>
Table date range end: date_to
Max. year: 2199

Click on SQL
to generate SQL statement
to create `dim_product`
table

Generate SQL

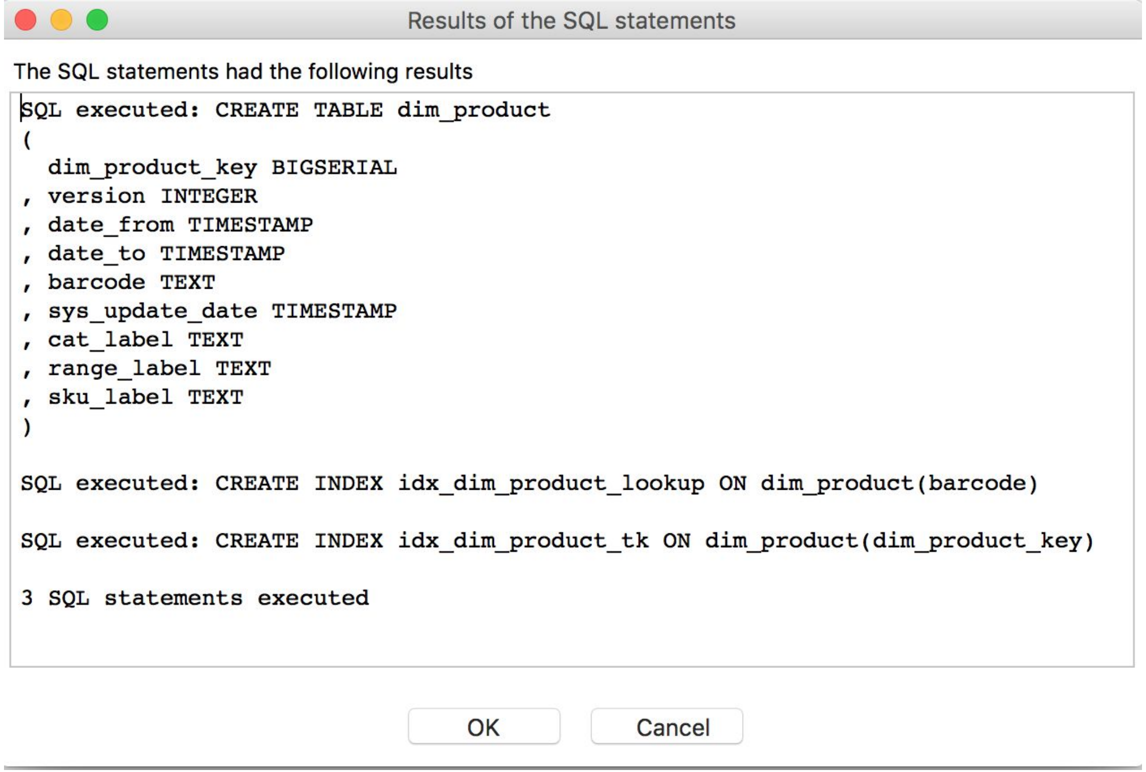


SQL statements, separated by semicolon ';'

```
CREATE TABLE dim_product
(
  dim_product_key BIGSERIAL
, version INTEGER
, date_from TIMESTAMP
, date_to TIMESTAMP
, barcode TEXT
, sys_update_date TIMESTAMP
, cat_label TEXT
, range_label TEXT
, sku_label TEXT
)
;CREATE INDEX idx_dim_product_lookup ON dim_product(barcode)
;
CREATE INDEX idx_dim_product_tk ON dim_product(dim_product_key)
;
```

Line 1 column 0

Execute SQL



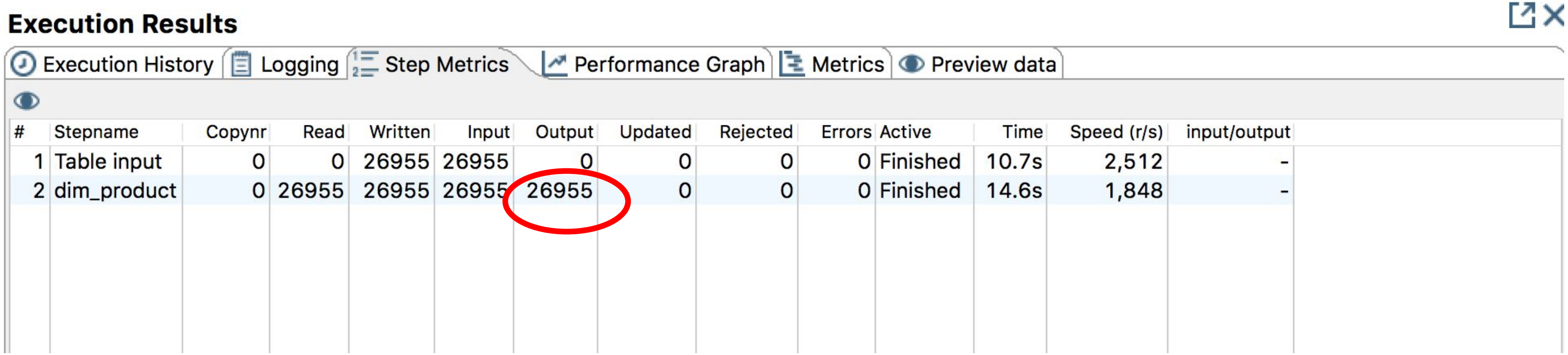
The SQL statements had the following results

```
SQL executed: CREATE TABLE dim_product
(
  dim_product_key BIGSERIAL
, version INTEGER
, date_from TIMESTAMP
, date_to TIMESTAMP
, barcode TEXT
, sys_update_date TIMESTAMP
, cat_label TEXT
, range_label TEXT
, sku_label TEXT
)

SQL executed: CREATE INDEX idx_dim_product_lookup ON dim_product(barcode)

SQL executed: CREATE INDEX idx_dim_product_tk ON dim_product(dim_product_key)

3 SQL statements executed
```

Task 1.2 - Run Transformation

- Run Transformation
- Watch step metrics
- Watch for errors in logging
- Run Transformation again
- Watch step metrics - should be no write rows

First Run

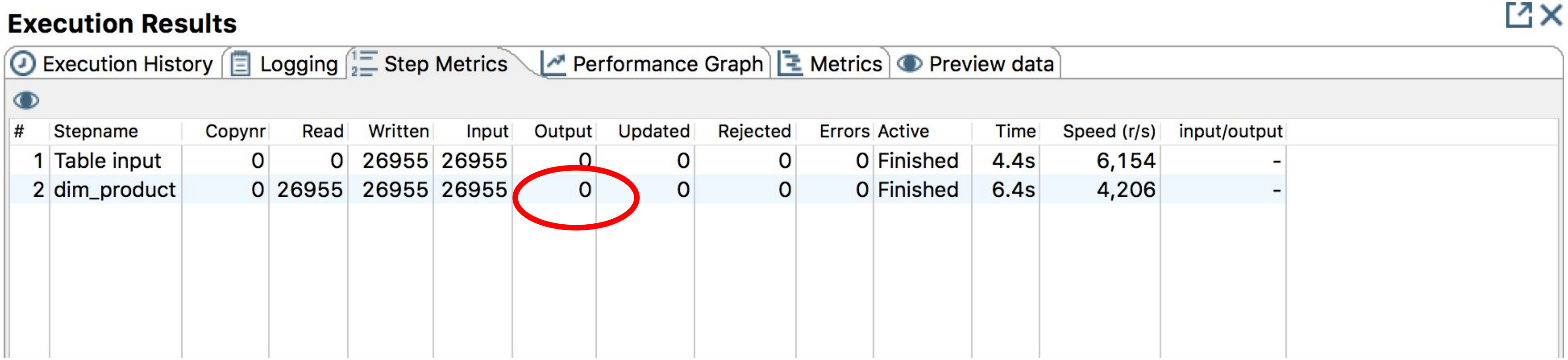


Execution Results

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	Table input	0	0	26955	26955	0	0	0	0	Finished	10.7s	2,512	-
2	dim_product	0	26955	26955	26955	26955	0	0	0	Finished	14.6s	1,848	-

Second Run

Second run and subsequent runs with same data will not insert new rows into the dimension table as it has to be unique



Execution Results

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	Table input	0	0	26955	26955	0	0	0	0	Finished	4.4s	6,154	-
2	dim_product	0	26955	26955	26955	0	0	0	0	Finished	6.4s	4,206	-

DIY

1. Try adding the following row in products csv file manually:

1010000007,10,FACIAL SKIN CARE/1,100,VITAMIN D,10,VIT.D SKIN BOOST 30ML,UNKNOWN,,,1,1,1,Centimeter,Grams,98

2. Run the stage_product transf.

3. Run the dim_product transf. and Notice the step metrics

   a. What do you see in Output field of dim_product? And why it is so?

4. Now remove that line from the csv file. Then run stage_product, dim_product transformations. Notice the output field in step metrics. Explain what you see.

*Hint: It has related to the “punch throw” option.

End of Task 1.2