

Data Analysis and Integration

ETL tools and SQL queries

Pentaho Data Integration

- Transformations work in “streaming” mode
 - the first row could reach the output before the second row is read from input
 - there are exceptions, e.g. when some aggregation on multiple rows needs to be performed
- Transformations can be saved and executed many times
- Transformations can become quite complex

SQL query

- number of employees in each department

emp_no	dept_no
10721	d009
11260	d009
11371	d005
11693	d005
13816	d005
14007	d002
14083	d004
14791	d005
17698	d005
17739	d005
...	...

dept_no	dept_name
d009	Customer Service
d005	Development
d002	Finance
d003	Human Resources
d001	Marketing
d004	Production
d006	Quality Management
d008	Research
d007	Sales

SQL query

- number of employees in each department

```
select *  
from curr_dept_emp as a, departments as b  
where a.dept_no = b.dept_no;
```

emp_no	dept_no	dept_no	dept_name
10721	d009	d009	Customer Service
11260	d009	d009	Customer Service
11371	d005	d005	Development
11693	d005	d005	Development
13816	d005	d005	Development
14007	d002	d002	Finance
14083	d004	d004	Production
14791	d005	d005	Development
17698	d005	d005	Development
17739	d005	d005	Development

10 rows in set (0.00 sec)

SQL query

- number of employees in each department

```
select b.dept_no, b.dept_name, count(emp_no)
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by b.dept_no, b.dept_name;
```

dept_no	dept_name	count(emp_no)
d001	Marketing	15
d002	Finance	18
d003	Human Resources	10
d004	Production	44
d005	Development	62
d006	Quality Management	18
d007	Sales	42
d008	Research	14
d009	Customer Service	29

9 rows in set (0.00 sec)

SQL query

- number of employees in each department

```
select b.dept_no, b.dept_name, count(emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by b.dept_no, b.dept_name
having count_emp_no >= 40;
```

dept_no	dept_name	count_emp_no
d004	Production	44
d005	Development	62
d007	Sales	42

3 rows in set (0.00 sec)

SQL query

- number of employees in each department

```
select b.dept_no, b.dept_name, count(emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by b.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```

```
+-----+-----+-----+
| dept_no | dept_name | count_emp_no |
+-----+-----+-----+
| d005    | Development | 62 |
| d004    | Production  | 44 |
| d007    | Sales       | 42 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

SQL query

- doing the same query with an ETL tool

```
select b.dept_no, b.dept_name, count(emp_no) as count_emp_no  
from curr_dept_emp as a, departments as b  
where a.dept_no = b.dept_no  
group by b.dept_no, b.dept_name  
having count_emp_no >= 40  
order by count_emp_no desc;
```

– how to do it?

Table input

```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



Table input

Step name

Connection

SQL

```
SELECT
  emp_no
, dept_no
FROM curr_dept_emp
```

Table input

```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



Table input

Step name

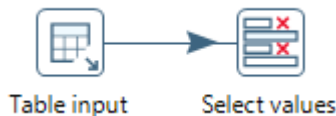
Connection

SQL

```
SELECT
  dept_no
, dept_name
FROM departments
```

Select values

```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



Select / Rename values

Step name:

Select & Alter Remove Meta-data

Fields:

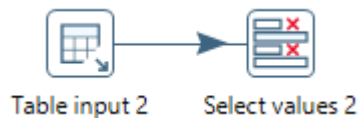
#	Fieldname	Rename to	Length	Precision
1	emp_no			
2	dept_no	dept_no_1		

Get fields to select
Edit Mapping

Include unspecified fields, ☐

Select values

```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



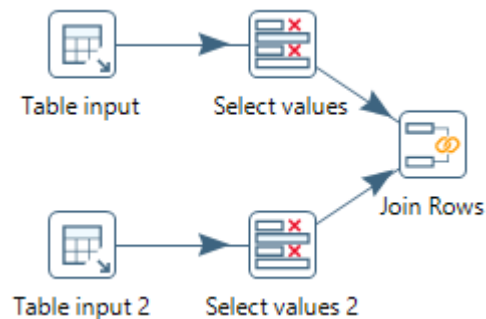
The screenshot shows a software window titled 'Select / Rename values'. At the top, there is a 'Step name' field containing the text 'Select values 2'. Below this, there are three tabs: 'Select & Alter' (which is active), 'Remove', and 'Meta-data'. Under the 'Select & Alter' tab, there is a section labeled 'Fields:' containing a table with the following data:

#	Fieldname	Rename to	Length	Precision
1	dept_no	dept_no_2		
2	dept_name			

A red arrow points to the 'dept_no_2' value in the 'Rename to' column. To the right of the table are two buttons: 'Get fields to select' and 'Edit Mapping'. At the bottom of the window, there is a checkbox labeled 'Include unspecified fields,' which is currently unchecked.

Join rows

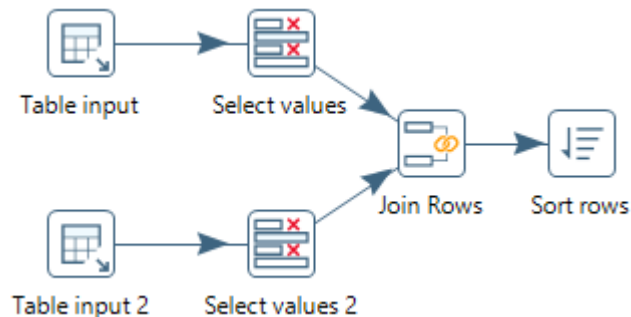
```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



The screenshot shows the 'Join rows' configuration window. The 'Step name' is 'Join Rows'. The 'Temp directory' is set to '%java.io.tmpdir%' with a 'Browse...' button. The 'TMP-file prefix' is 'out'. The 'Max. cache size (in rows)' is '500'. The 'Main step to read from' is a dropdown menu. The 'The condition:' section shows a join condition: 'dept_no_1' followed by an equals sign and 'dept_no_2'. Red arrows point to these fields.

Sort rows

```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



Sort rows

Step name: Sort rows

Sort directory: %%java.io.tmpdir%%

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

Compress TMP Files? ☐

Only pass unique rows? (verifies keys only) ☐

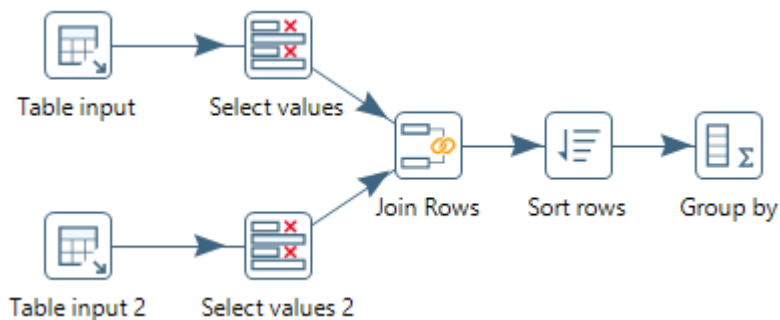
Fields:

#	Fieldname	Ascending	Case sensitive compare?
1	dept_no_1	Y	
2	dept_name	Y	
<			

Help OK Cancel

Group by

```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



Group By

Step name:

Include all rows? ☐

Temporary files directory:

TMP-file prefix:

Add line number, restart in each group ☐

Line number field name:

Always give back a result row ☐

The fields that make up the group:

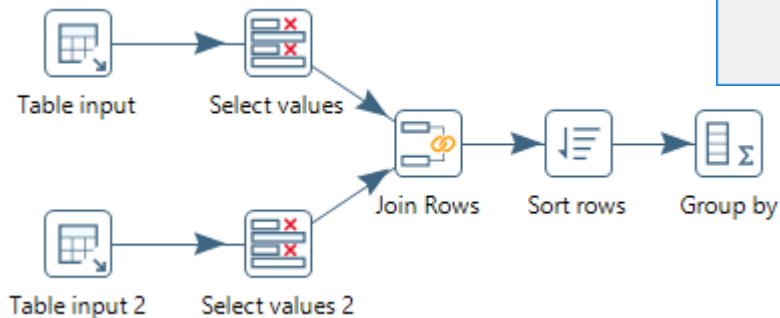
#	Group field
1	dept_no_1
2	dept_name

Aggregates:

#	Name	Subject	Type
1	count_emp_no	emp_no	Number of Values (N)

Group by

```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



Group By

Warning!

The group by function needs the input to be sorted on the specified keys. If you don't sort the input, the results may not be correct

☐ Please, don't show this warning anymore.

I understand

The fields that make up the group:

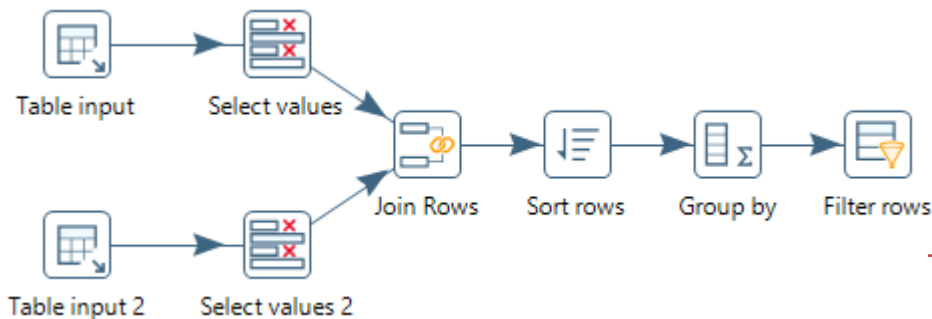
#	Group field	
1	dept_no_1	←
2	dept_name	←

Aggregates :

#	Name	Subject	Type
1	count_emp_no	emp_no	Number of Values (N)

Filter rows

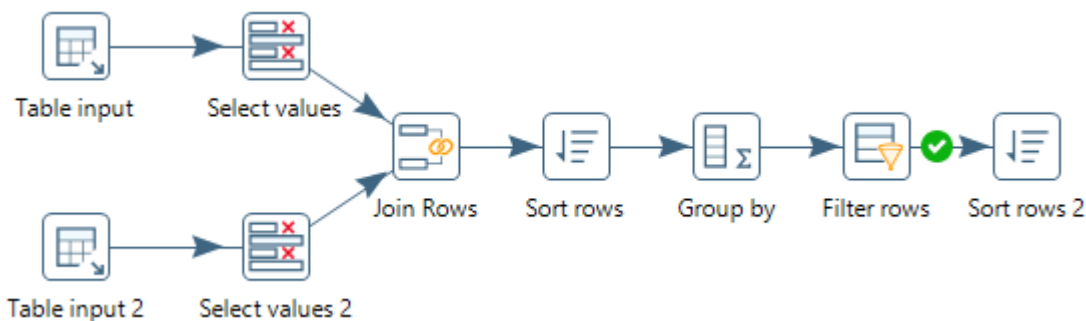
```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



The screenshot shows the configuration window for the 'Filter rows' step. The 'Step name' is set to 'Filter rows'. Below this, there are two dropdown menus for 'Send 'true' data to step:' and 'Send 'false' data to step:'. The 'The condition:' section is expanded, showing a configuration for the condition: `count_emp_no >= 40 (Integer)`. A red arrow points to the '40' value in the condition.

Filter rows

```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```



Sort rows

Step name: Sort rows 2

Sort directory: %%java.io.tmpdir%%

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

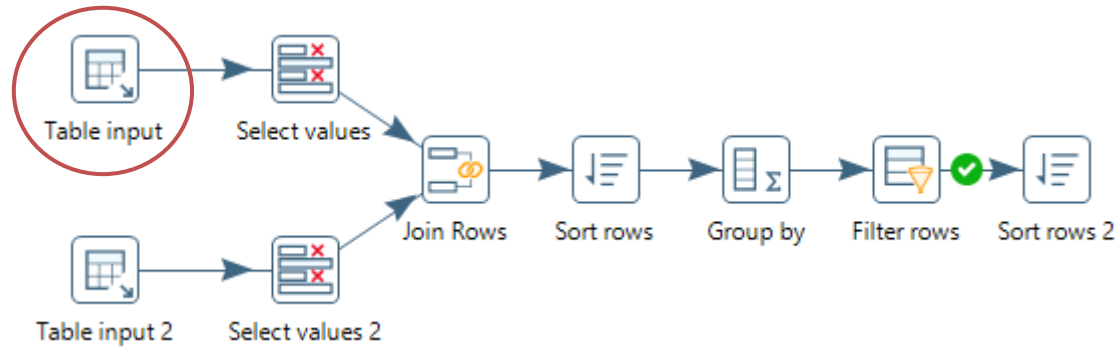
Compress TMP Files? ☐

Only pass unique rows? (verifies keys only) ☐

Fields:

#	Fieldname	Ascending	Case sensitive compare?
1	count_emp_no	N	

Preview



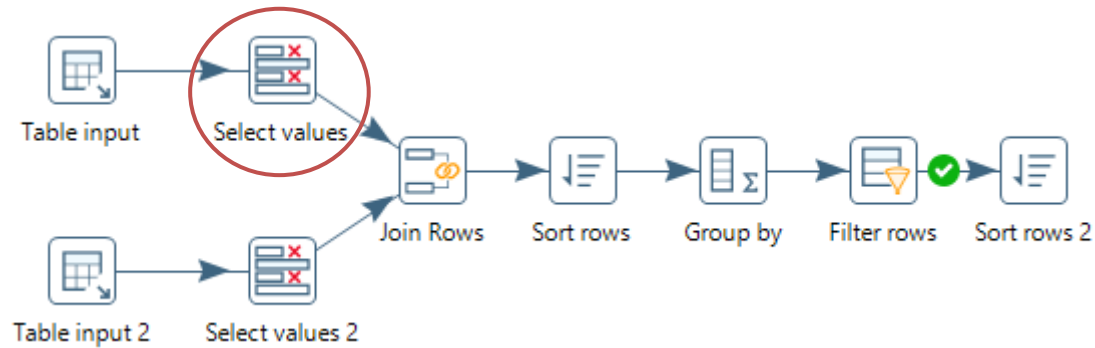
Examine preview data


Rows of step: Table input (252 rows)

#	emp_no	dept_no
1	10721	d009
2	11260	d009
3	11371	d005
4	11693	d005
5	13816	d005
6	14007	d002
7	14083	d004
8	14791	d005

Close

Preview



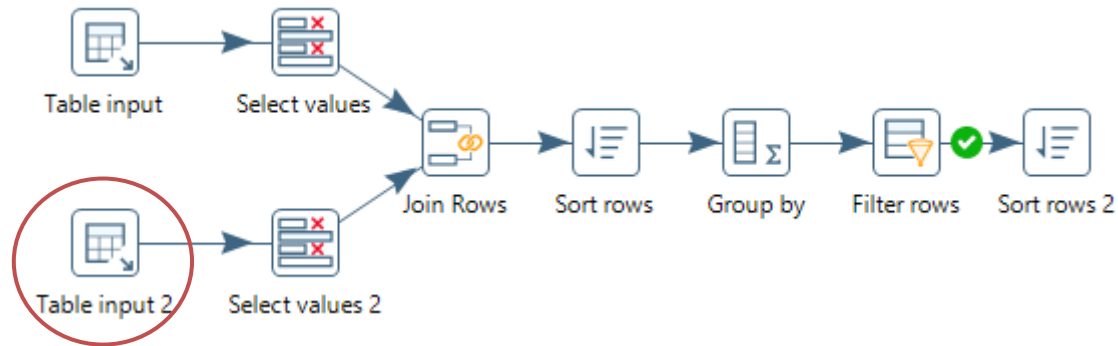
 Examine preview data

Rows of step: Select values (252 rows)

#	emp_no	dept_no_1
1	10721	d009
2	11260	d009
3	11371	d005
4	11693	d005
5	13816	d005
6	14007	d002
7	14083	d004
8	14791	d005

Close

Preview



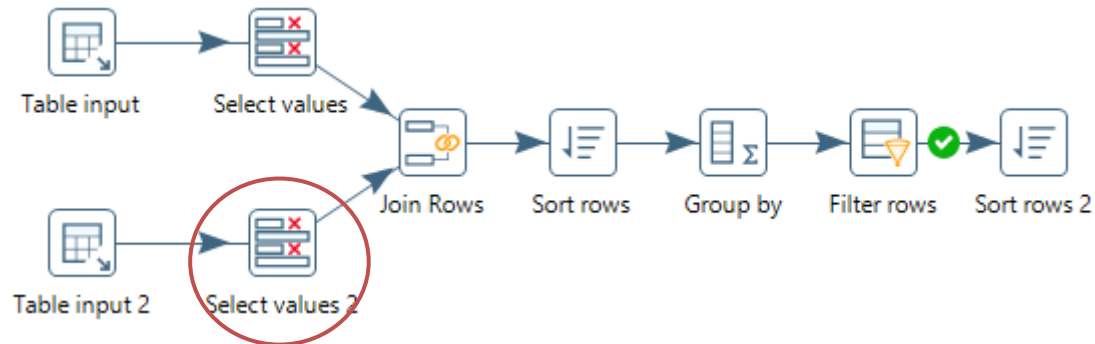
Examine preview data

Rows of step: Table input 2 (9 rows)

#	dept_no	dept_name
1	d009	Customer Service
2	d005	Development
3	d002	Finance
4	d003	Human Resources
5	d001	Marketing
6	d004	Production
7	d006	Quality Management
8	d008	Research

Close

Preview



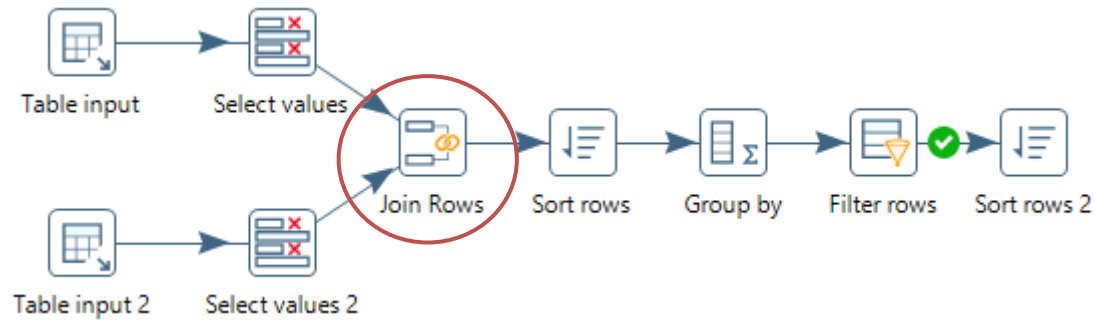
Examine preview data

Rows of step: Select values 2 (9 rows)

#	dept_no_2	dept_name
1	d009	Customer Service
2	d005	Development
3	d002	Finance
4	d003	Human Resources
5	d001	Marketing
6	d004	Production
7	d006	Quality Management
8	d008	Research

Close

Preview



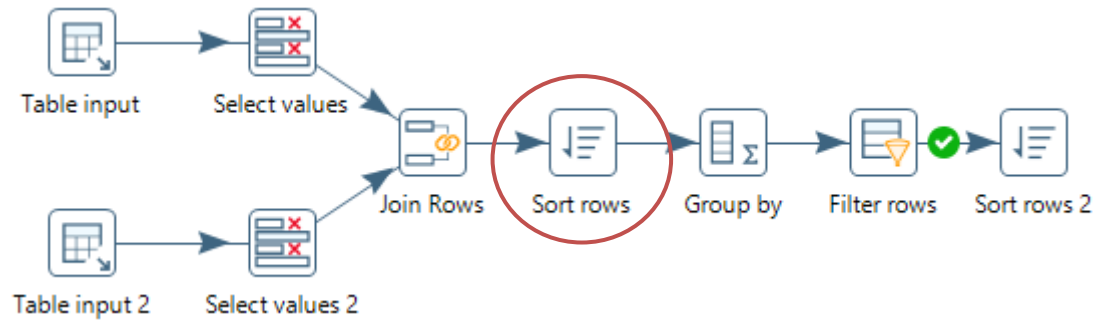
Examine preview data

Rows of step: Join Rows (252 rows)

#	emp_no	dept_no_1	dept_no_2	dept_name
1	10721	d009	d009	Customer Service
2	11260	d009	d009	Customer Service
3	11371	d005	d005	Development
4	11693	d005	d005	Development
5	13816	d005	d005	Development
6	14007	d002	d002	Finance
7	14083	d004	d004	Production
8	14791	d005	d005	Development

Close

Preview



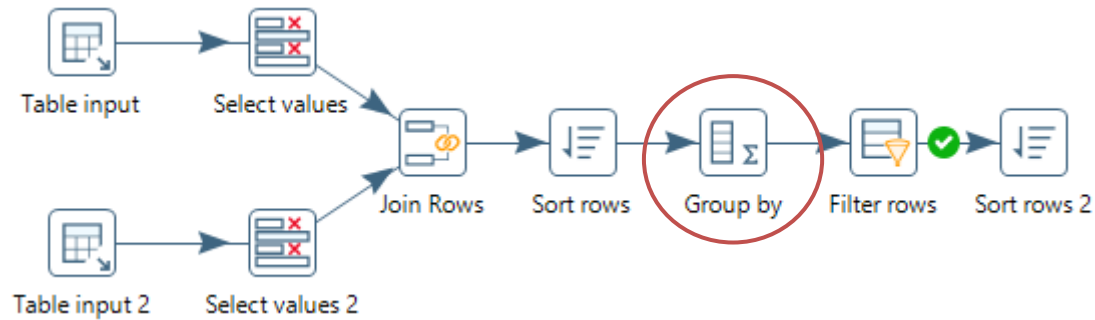
Examine preview data

Rows of step: Sort rows (252 rows)

#	emp_no	dept_no_1	dept_no_2	dept_name
1	21637	d001	d001	Marketing
2	25949	d001	d001	Marketing
3	44474	d001	d001	Marketing
4	84372	d001	d001	Marketing
5	102629	d001	d001	Marketing
6	104349	d001	d001	Marketing
7	110022	d001	d001	Marketing
8	110039	d001	d001	Marketing

Close

Preview



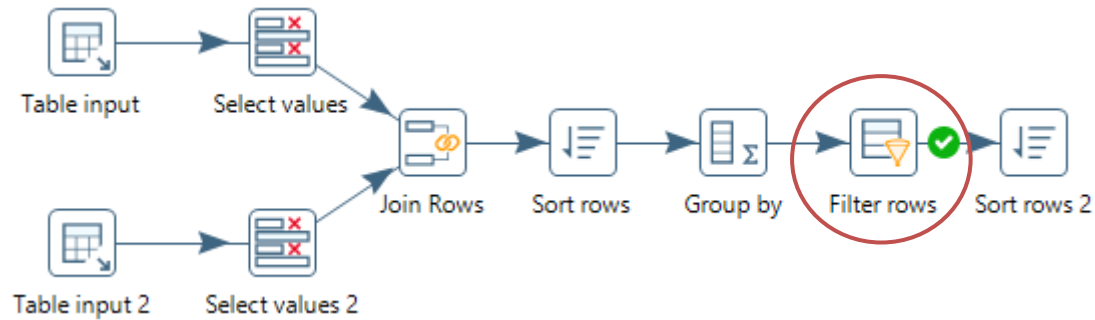
Examine preview data

Rows of step: Group by (9 rows)

#	dept_no_1	dept_name	count_emp_no
1	d001	Marketing	15
2	d002	Finance	18
3	d003	Human Resources	10
4	d004	Production	44
5	d005	Development	62
6	d006	Quality Management	18
7	d007	Sales	42
8	d008	Research	14
9	d009	Customer Service	29

Close

Preview



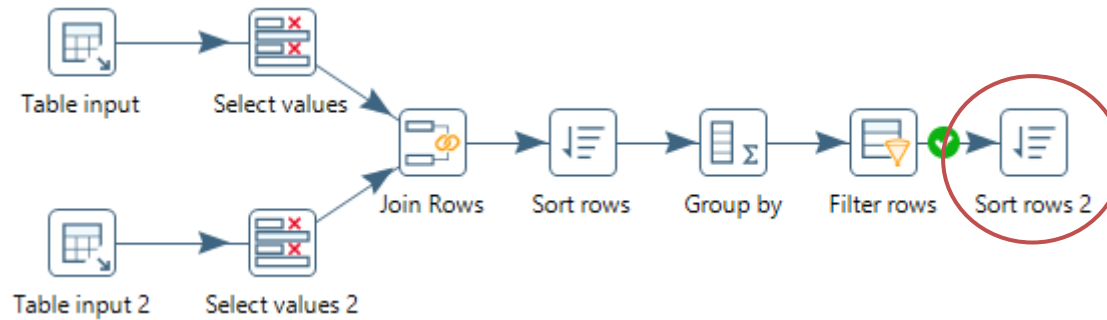
Examine preview data

Rows of step: Filter rows (3 rows)

#	dept_no_1	dept_name	count_emp_no
1	d004	Production	44
2	d005	Development	62
3	d007	Sales	42

Close

Preview



Examine preview data

Rows of step: Sort rows 2 (3 rows)

#	dept_no_1	dept_name	count_emp_no
1	d005	Development	62
2	d004	Production	44
3	d007	Sales	42

Close

Alternatives

- A completely different solution



Table input

Step name: Table input

Connection: employees [Edit...] [New...] [Wizard...]

SQL [Get SQL select statement...]

```
select a.dept_no, b.dept_name, count(a.emp_no) as count_emp_no
from curr_dept_emp as a, departments as b
where a.dept_no = b.dept_no
group by a.dept_no, b.dept_name
having count_emp_no >= 40
order by count_emp_no desc;
```

Line 7 Column 0

Enable lazy conversion ☐

Replace variables in script? ☐

Insert data from step [v]

Execute for each row? ☐

Limit size: 0

[?] Help [OK] [Preview] [Cancel]

Alternatives

- A completely different solution



Examine preview data

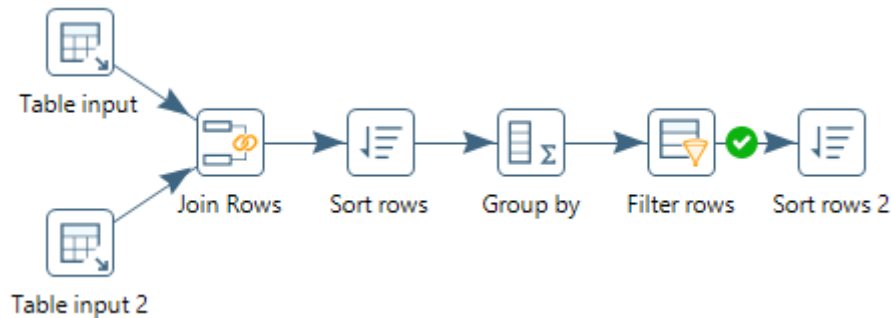
Rows of step: Table input (3 rows)

#	dept_no	dept_name	count_emp_no
1	d005	Development	62
2	d004	Production	44
3	d007	Sales	42

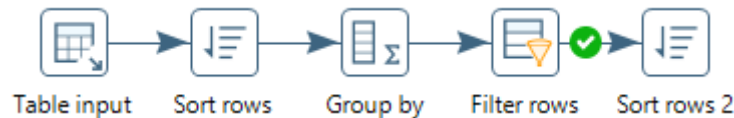
Close

Alternatives

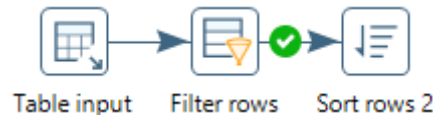
- Other possible intermediate solutions
 - column renaming in SQL



- column renaming and table join in SQL



- column renaming, table join, and group by in SQL



Alternatives

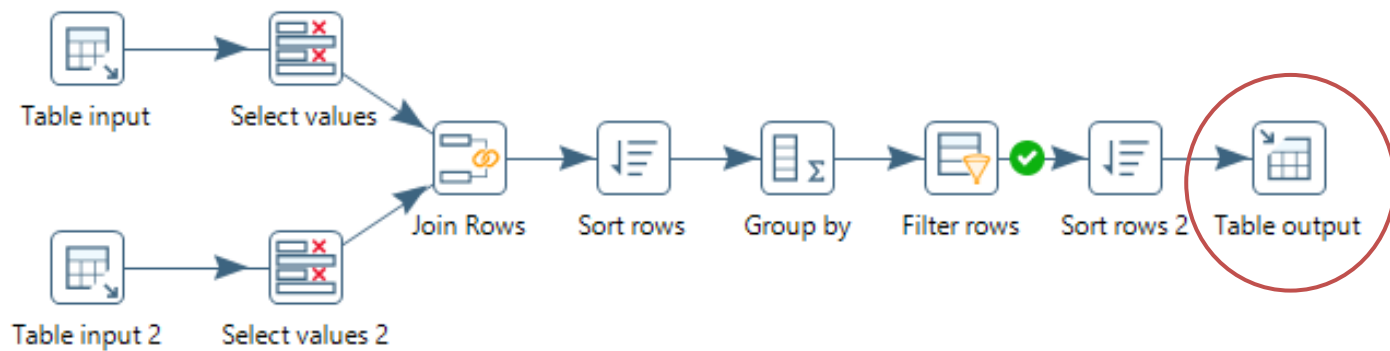
- As a general rule
 - offload as much as possible to database system
 - filtering rows, selecting columns, and (especially) table joins
 - unless the required data comes from multiple data sources
 - or when processing is difficult to implement in SQL
 - e.g. duplicate detection via string matching

ETL tools

- Then why use an ETL tool?
 - data comes from different databases/systems
 - data sources other than databases (e.g. text files)
 - complex data merging and transformations
 - approximate matching, duplicate detection, data cleaning
 - materialization to different outputs (databases, files, etc.)

Materialization

- Materialization to database table



Materialization

- Materialization to database table

Table output

Step name: Table output

Connection: employees [Edit...] [New...] [Wizard...]

Target schema: [Browse...]

Target table: num_emp_dept [Browse...] ←

Commit size: 1000

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☐

Main options | **Database fields**

Partition data over tables: ☐

Partitioning field: [v]

Partition data per month: ☒

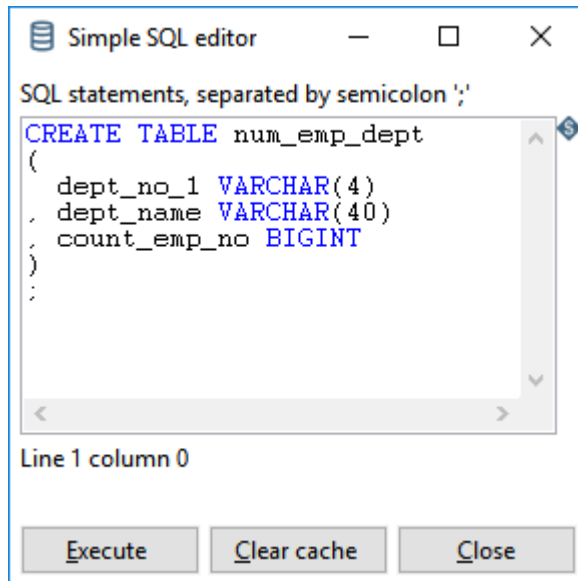
Partition data per day: ☐

Use batch update for inserts: ☒

[?] Help [OK] [Cancel] **SQL**

Materialization

- Materialization to database table

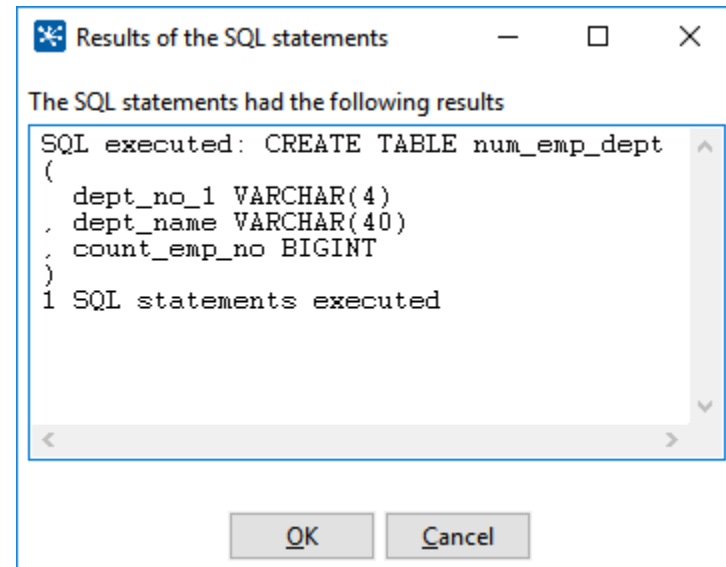


A screenshot of a 'Simple SQL editor' window. The title bar includes a document icon, the text 'Simple SQL editor', and standard window controls (minimize, maximize, close). The main text area contains the SQL statement: `CREATE TABLE num_emp_dept (dept_no_1 VARCHAR(4) , dept_name VARCHAR(40) , count_emp_no BIGINT) ;`. The text is color-coded: 'CREATE TABLE' is blue, 'num_emp_dept' is black, and the column names and data types are blue. Below the text area, it says 'Line 1 column 0'. At the bottom, there are three buttons: 'Execute', 'Clear cache', and 'Close'.

```
CREATE TABLE num_emp_dept
(
  dept_no_1 VARCHAR(4)
, dept_name VARCHAR(40)
, count_emp_no BIGINT
)
;
```

Line 1 column 0

Execute Clear cache Close



A screenshot of a 'Results of the SQL statements' window. The title bar includes a SQL icon, the text 'Results of the SQL statements', and standard window controls. The main text area displays the results of the executed SQL statement: 'SQL executed: CREATE TABLE num_emp_dept (dept_no_1 VARCHAR(4) , dept_name VARCHAR(40) , count_emp_no BIGINT) 1 SQL statements executed'. At the bottom, there are two buttons: 'OK' and 'Cancel'.

The SQL statements had the following results

```
SQL executed: CREATE TABLE num_emp_dept
(
  dept_no_1 VARCHAR(4)
, dept_name VARCHAR(40)
, count_emp_no BIGINT
)
1 SQL statements executed
```

OK Cancel

Materialization

- Running the transformation

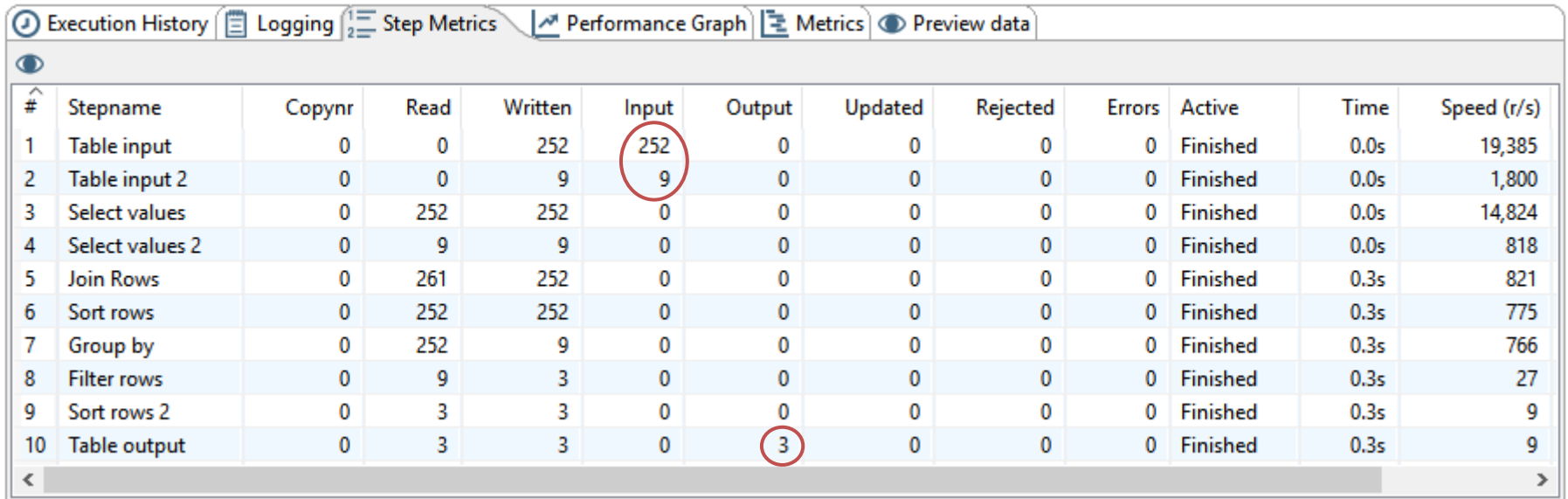
The screenshot displays the Spoon - transformation02 window. The left sidebar shows a tree view of the transformation steps, including Input, Output, Transform, Utility, Flow, Scripting, Pentaho Server, Lookup, Joins, Data Warehouse, Validation, Statistics, Big Data, Agile, Cryptography, Palo, Open ERP, Job, Mapping, Bulk loading, Inline, and Experimental. The main canvas shows a workflow diagram with the following steps: Table input, Select values, Join Rows, Sort rows, Group by, Filter rows, Sort rows 2, and Table output. A red circle highlights the 'Run' button in the top toolbar. Below the workflow diagram, the 'Execution Results' section is visible, showing a table with the following data:

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active
1	Table input	0	0	252	252	0	0	0	0	Finished
2	Table input 2	0	0	9	9	0	0	0	0	Finished
3	Select values	0	252	252	0	0	0	0	0	Finished
4	Select values 2	0	9	9	0	0	0	0	0	Finished
5	Join Rows	0	261	252	0	0	0	0	0	Finished
6	Sort rows	0	252	252	0	0	0	0	0	Finished
7	Group by	0	252	9	0	0	0	0	0	Finished
8	Filter rows	0	9	3	0	0	0	0	0	Finished
9	Sort rows 2	0	3	3	0	0	0	0	0	Finished
10	Table output	0	3	3	0	3	0	0	0	Finished

Materialization

- Running the transformation

Execution Results



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

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	Table input	0	0	252	252	0	0	0	0	Finished	0.0s	19,385
2	Table input 2	0	0	9	9	0	0	0	0	Finished	0.0s	1,800
3	Select values	0	252	252	0	0	0	0	0	Finished	0.0s	14,824
4	Select values 2	0	9	9	0	0	0	0	0	Finished	0.0s	818
5	Join Rows	0	261	252	0	0	0	0	0	Finished	0.3s	821
6	Sort rows	0	252	252	0	0	0	0	0	Finished	0.3s	775
7	Group by	0	252	9	0	0	0	0	0	Finished	0.3s	766
8	Filter rows	0	9	3	0	0	0	0	0	Finished	0.3s	27
9	Sort rows 2	0	3	3	0	0	0	0	0	Finished	0.3s	9
10	Table output	0	3	3	0	3	0	0	0	Finished	0.3s	9

Materialization

- Materialization to database table

```
select * from num_emp_dept;
```

```
+-----+-----+-----+
| dept_no_1 | dept_name | count_emp_no |
+-----+-----+-----+
| d005      | Development | 62 |
| d004      | Production  | 44 |
| d007      | Sales      | 42 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```