Content

Name: Chiyu Chen **StudentID:** 901265
Name: Jie Niu **StudentID:** 890649

1 Executive summary

1.1 Content summary

This report mainly focuses on the design of ETL (Extract, Transform and Load) process, which aims to integrate the chaotic, scattered and inconsistent data in business and provides the analytical basis for the business decision. Additionally, the redesign of data warehouse is discussed in this report as well for proper storage. In the end of report, it represents the data dictionary that describes types of meta-data.

1.2 Issues addressed in ETL process

There are three issues met in the process of ETL design.

The first primary issue is scattered and chaotic types of data from original data systems. Since the different system choices and inconsistent data representations, to control data quality and yield uniform and high-quality data is the main topic in ETL process. It can reduce the dispensable workload significantly.

The second problem is to deal with the unvalued data such as null-value data. During the transform procedure, the Pentaho reports errors due to existence of null-value data. Deleting this kind of data can reduce the unnecessary computation and save storage space.

The last issue that needs to be overcame is how to address the Slowly Changing Dimension (SCD). According to Kimball's methods, this kind of dimension needs to store both current and historical data over time. And the method of Type 2 is what we apply in this report. In short, the surrogate key is applied in this case. This method is clear and powerful. It maintains the entire historical record.

2 Design of the ETL Process

2.1 Time Dimensional Table Transformation

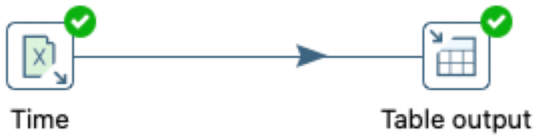
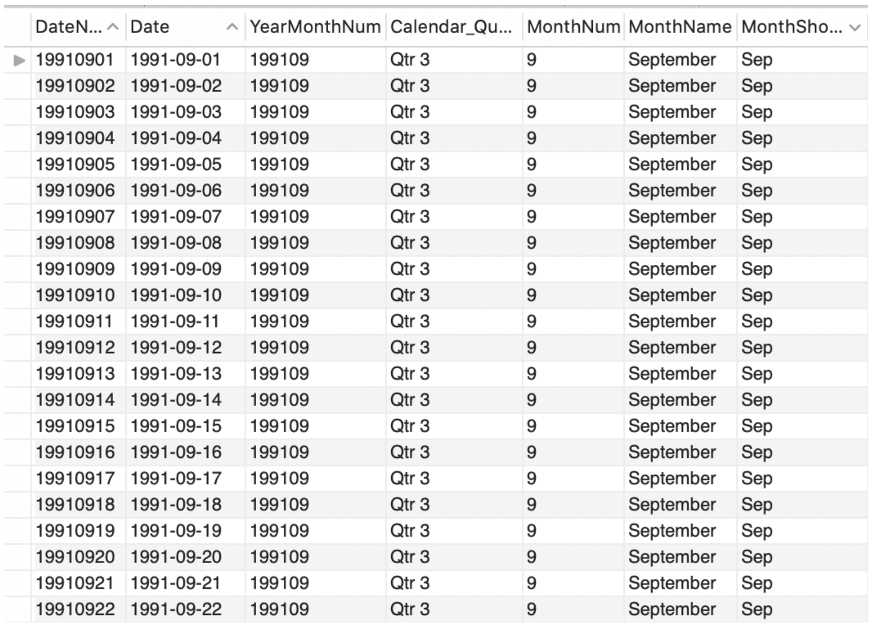


Figure 1. Time Transformation

In this time transformation, it can be achieved by two steps. The first one is to input the *DimDates.xlsx* file and the second step is to create the relative table and output the data from source file to MySQL. The part of output as below:



	DateN...	Date	YearMonthNum	Calendar_Qu...	MonthNum	MonthName	MonthSho...
▶	19910901	1991-09-01	199109	Qtr 3	9	September	Sep
	19910902	1991-09-02	199109	Qtr 3	9	September	Sep
	19910903	1991-09-03	199109	Qtr 3	9	September	Sep
	19910904	1991-09-04	199109	Qtr 3	9	September	Sep
	19910905	1991-09-05	199109	Qtr 3	9	September	Sep
	19910906	1991-09-06	199109	Qtr 3	9	September	Sep
	19910907	1991-09-07	199109	Qtr 3	9	September	Sep
	19910908	1991-09-08	199109	Qtr 3	9	September	Sep
	19910909	1991-09-09	199109	Qtr 3	9	September	Sep
	19910910	1991-09-10	199109	Qtr 3	9	September	Sep
	19910911	1991-09-11	199109	Qtr 3	9	September	Sep
	19910912	1991-09-12	199109	Qtr 3	9	September	Sep
	19910913	1991-09-13	199109	Qtr 3	9	September	Sep
	19910914	1991-09-14	199109	Qtr 3	9	September	Sep
	19910915	1991-09-15	199109	Qtr 3	9	September	Sep
	19910916	1991-09-16	199109	Qtr 3	9	September	Sep
	19910917	1991-09-17	199109	Qtr 3	9	September	Sep
	19910918	1991-09-18	199109	Qtr 3	9	September	Sep
	19910919	1991-09-19	199109	Qtr 3	9	September	Sep
	19910920	1991-09-20	199109	Qtr 3	9	September	Sep
	19910921	1991-09-21	199109	Qtr 3	9	September	Sep
	19910922	1991-09-22	199109	Qtr 3	9	September	Sep

Figure 2. Part of Time transformation output

2.2 Market Dimensional Table Transformation

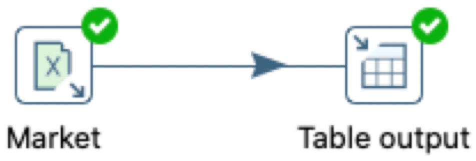
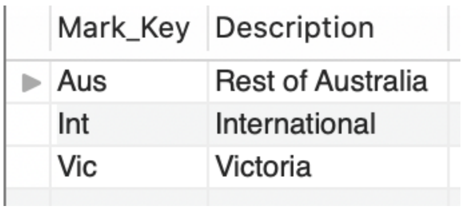


Figure 3. Market Transformation

In the market transformation, the original design of data warehouse can meet the file *Market.xlsx* from source data system. Therefore, it can be transformed directly to data warehouse. The first step is to input source file *Market.xlsx* and the second one is to create a new relative table and output data to it.

The partial result of output as follow:



	Mark_Key	Description
▶	Aus	Rest of Australia
	Int	International
	Vic	Victoria

Figure 4. Part of Market transformation output

2.3 Agent Dimensional Table Transformation

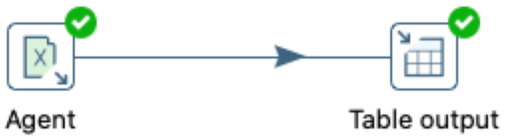
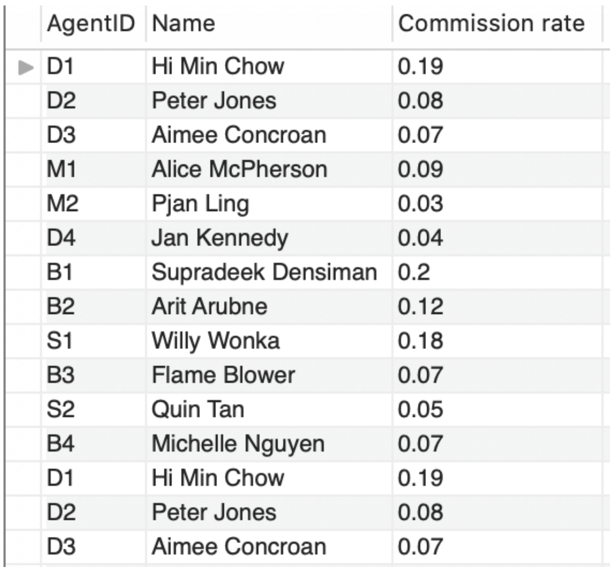


Figure 5. Agent Transformation

As shows in the Figure 5, the agent transformation has two steps, one is to input the data source file *SalesAgent.xlsx* to Pentaho, and the other is to output the result of this process to data warehouse system.

The final result of this transformation is shown as below:



AgentID	Name	Commission rate
D1	Hi Min Chow	0.19
D2	Peter Jones	0.08
D3	Aimee Concroan	0.07
M1	Alice McPherson	0.09
M2	Pjan Ling	0.03
D4	Jan Kennedy	0.04
B1	Supradeek Densiman	0.2
B2	Arit Arubne	0.12
S1	Willy Wonka	0.18
B3	Flame Blower	0.07
S2	Quin Tan	0.05
B4	Michelle Nguyen	0.07
D1	Hi Min Chow	0.19
D2	Peter Jones	0.08
D3	Aimee Concroan	0.07

Figure 6. Part of Agent transformation output

2.4 Product Dimensional Table Transformation

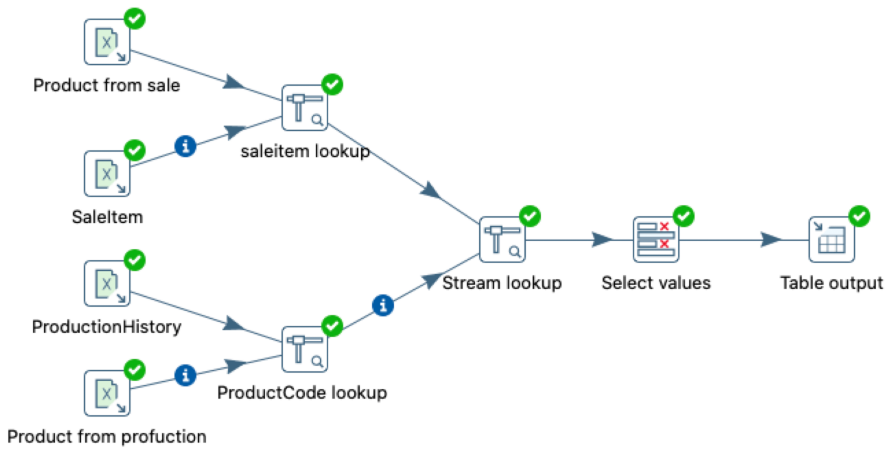
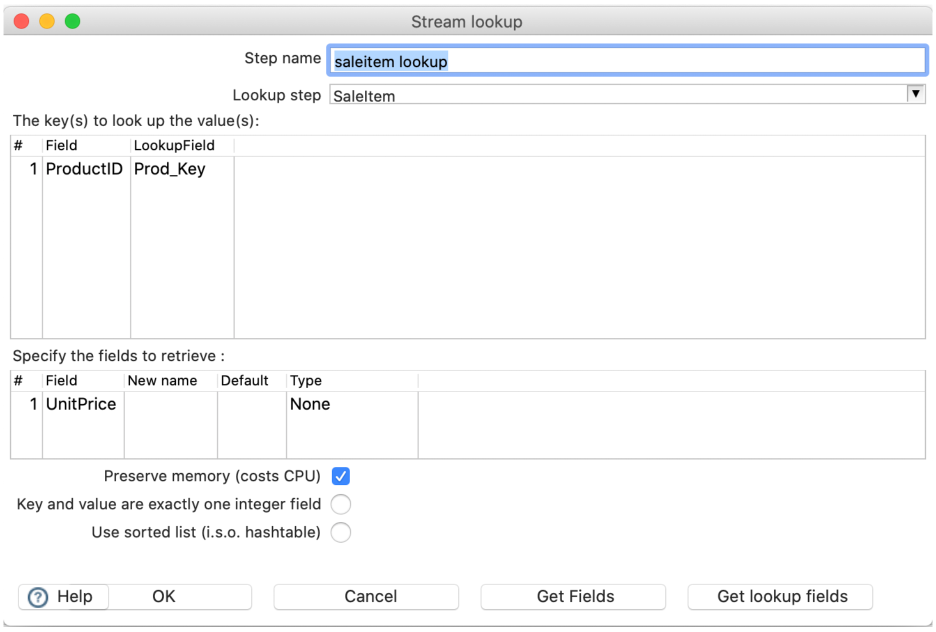


Figure 7. Product Transformation

In the product dimensional table transformation, it needs four files from both two original systems as source data, which are *Product.xlsx* in SalesSystem marked as ‘Product from sale’, *SaleItem.xlsx* and *ProductionHistory.xlsx* and *Product.xlsx* in ProductionSystem marked as ‘Product from production’.

The ‘Product from sale’ provides the major structure of this dimensional table. For example, the ‘ProductKey’ in this file is mapped as ProductID, and the ‘Price’ is modified as AdvisedPrice which is more specific and clearer. ‘Description’ and ‘Group’ are used as the same name and date format. Also, the ‘ProdYear’ is deleted as it is seen as a redundant data to product-relative information. This process is achieved by the second last step called ‘Select values’.

The reason for applying *SaleItem.xlsx* is to provide the unit price for each product in product dimension table. The step to complete it is by ‘saleitem lookup’. The details in this step is shown below:



Step name: saleitem lookup

Lookup step: SaleItem

The key(s) to look up the value(s):

#	Field	LookupField
1	ProductID	Prod_Key

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	UnitPrice			None

Preserve memory (costs CPU) ✓

Key and value are exactly one integer field ○

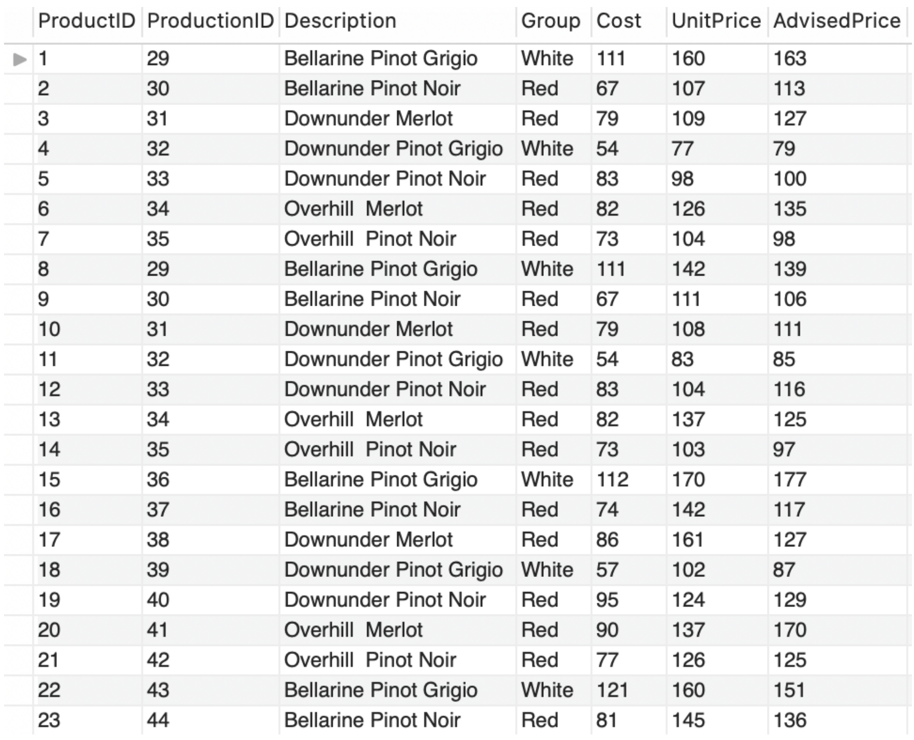
Use sorted list (i.s.o. hashtable) ○

Figure 8. saleitem lookup step of Product transformation

The input steps ‘ProductionHistory’ and ‘Product from production’ are used to offer the ProductionID and cost of each product. And this process is achieved by ‘ProductCode lookup’ step.

After the four files are ‘assembled’ separately in their systems, the ‘Stream lookup’ is to integrate them together. Finally, ‘Table output’ is to deliver the final result to data warehouse.

The partial final result is shown as below:

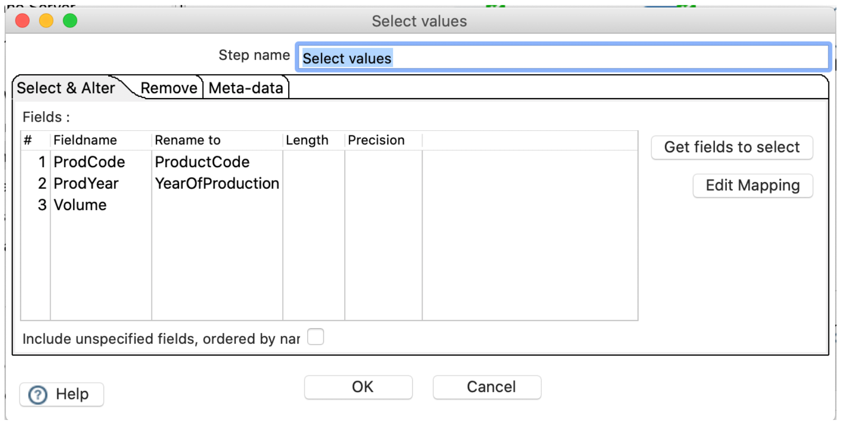


	ProductID	ProductionID	Description	Group	Cost	UnitPrice	AdvisedPrice
▶	1	29	Bellarine Pinot Grigio	White	111	160	163
	2	30	Bellarine Pinot Noir	Red	67	107	113
	3	31	Downunder Merlot	Red	79	109	127
	4	32	Downunder Pinot Grigio	White	54	77	79
	5	33	Downunder Pinot Noir	Red	83	98	100
	6	34	Overhill Merlot	Red	82	126	135
	7	35	Overhill Pinot Noir	Red	73	104	98
	8	29	Bellarine Pinot Grigio	White	111	142	139
	9	30	Bellarine Pinot Noir	Red	67	111	106
	10	31	Downunder Merlot	Red	79	108	111
	11	32	Downunder Pinot Grigio	White	54	83	85
	12	33	Downunder Pinot Noir	Red	83	104	116
	13	34	Overhill Merlot	Red	82	137	125
	14	35	Overhill Pinot Noir	Red	73	103	97
	15	36	Bellarine Pinot Grigio	White	112	170	177
	16	37	Bellarine Pinot Noir	Red	74	142	117
	17	38	Downunder Merlot	Red	86	161	127
	18	39	Downunder Pinot Grigio	White	57	102	87
	19	40	Downunder Pinot Noir	Red	95	124	129
	20	41	Overhill Merlot	Red	90	137	170
	21	42	Overhill Pinot Noir	Red	77	126	125
	22	43	Bellarine Pinot Grigio	White	121	160	151
	23	44	Bellarine Pinot Noir	Red	81	145	136

Figure 9. Part of Product transformation output

2.5 Production Dimensional Table Transformation

In this production transformation, to avoid the appearance of duplicate attributes in one data warehouse, the attributes selected in 'ProductionHistory' are ProdCode marked as ProductCode, ProdYear marked as YearOfProduction and volume. The 'Select values' is adopted to achieve this which is shown as below:



Step name: Select values

Fields :

#	Fieldname	Rename to	Length	Precision
1	ProdCode	ProductCode		
2	ProdYear	YearOfProduction		
3	Volume			

Include unspecified fields, ordered by nar ☐

Figure 10. Select values in Production Transformation

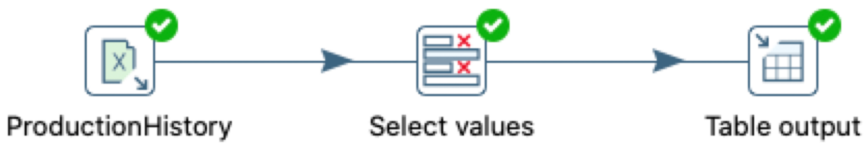
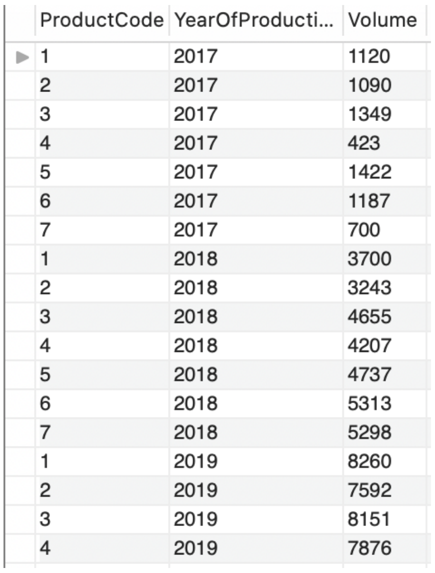


Figure 11. Production Transformation

The part of final output is shown as below:



ProductCode	YearOfProducti...	Volume
1	2017	1120
2	2017	1090
3	2017	1349
4	2017	423
5	2017	1422
6	2017	1187
7	2017	700
1	2018	3700
2	2018	3243
3	2018	4655
4	2018	4207
5	2018	4737
6	2018	5313
7	2018	5298
1	2019	8260
2	2019	7592
3	2019	8151
4	2019	7876

Figure 12. Part of Production transformation output

2.6 Sales Fact Table Transformation

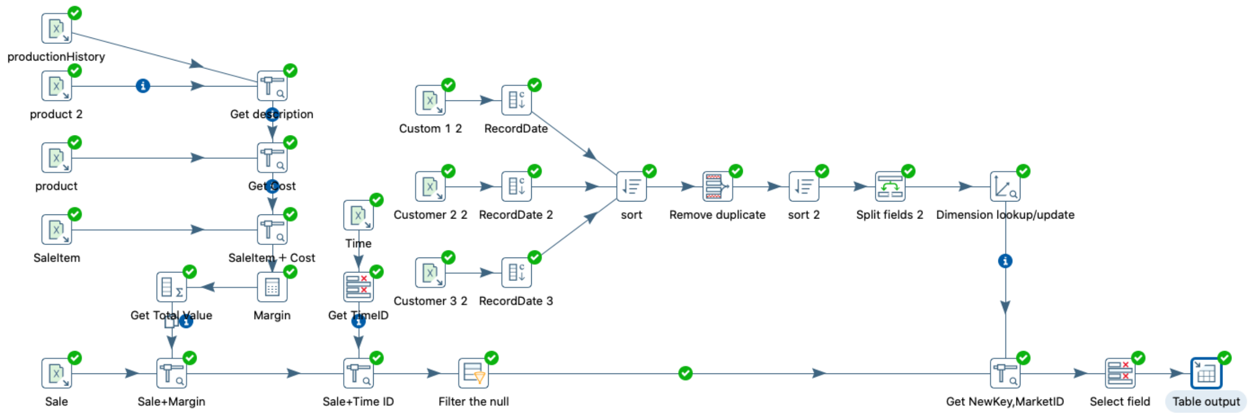
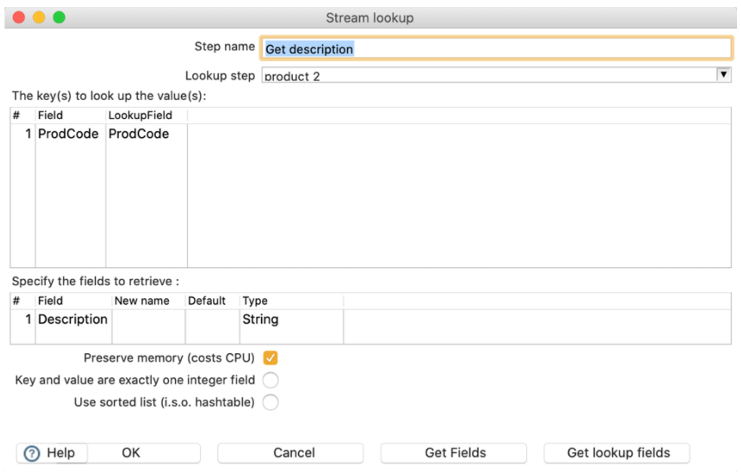


Figure 13. Sales Fact Table Transformation

In this transformation, based on the *sale* table in SalesSystem, the attributes required by the Sale fact table are provided to it through the relationship between the *sale* table and other tables.

The first thing that needs to be resolved is the acquisition of the Margin value. Because the same item may correspond to different production years, different costs are incurred. Therefore, the cost corresponding to each ProductKey in the *Product* table in SalesSystem needs to be matched.



Step name: Get description

Lookup step: product 2

The key(s) to look up the value(s):

#	Field	LookupField
1	ProdCode	ProdCode

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	Description			String

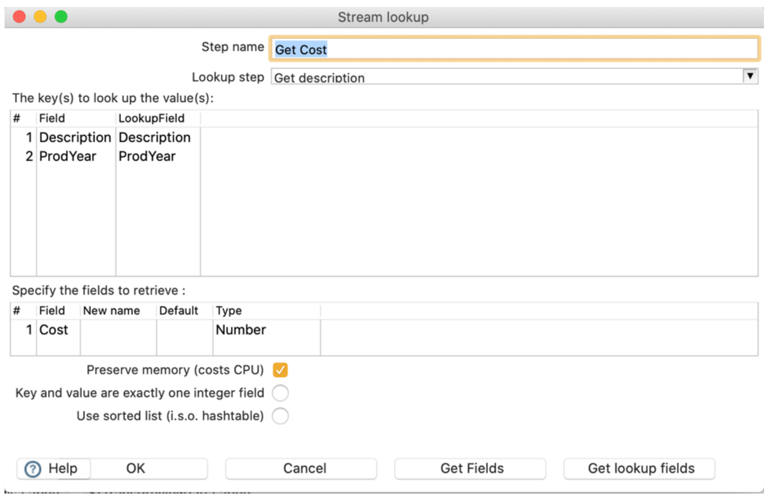
Preserve memory (costs CPU) ✓

Key and value are exactly one integer field

Use sorted list (i.s.o. hashtable)

Figure 14. Get description

The *Product* table in SalesSystem can query the corresponding cost from the *ProductionHistory* table by using the same Description value and ProdYear value.



Step name: Get Cost

Lookup step: Get description

The key(s) to look up the value(s):

#	Field	LookupField
1	Description	Description
2	ProdYear	ProdYear

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	Cost			Number

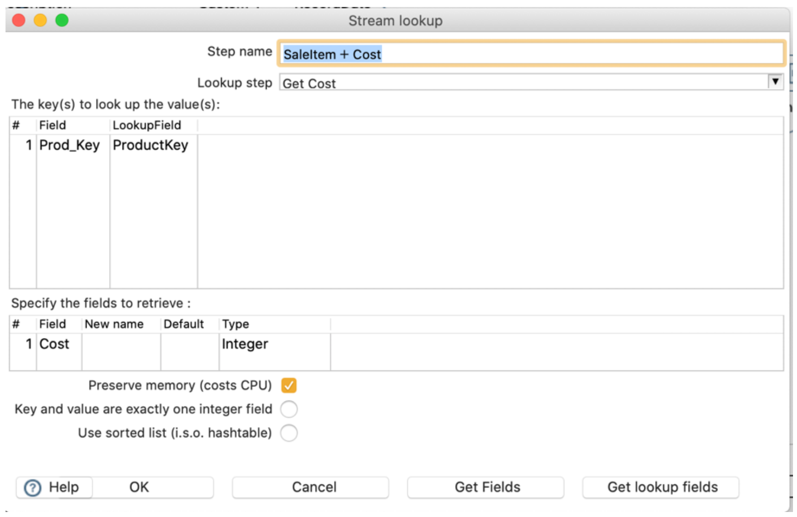
Preserve memory (costs CPU) ✓

Key and value are exactly one integer field

Use sorted list (i.s.o. hashtable)

Figure 15. Get cost

The *SaleItem* table has the transaction quantity and unit price of each item in each order, so by querying the value of Prod_Key, you can get the cost corresponding to each Prod_Key. Then you can calculate the benefit of each item in each order.


Stream lookup

Step name: SaleItem + Cost

Lookup step: Get Cost

The key(s) to look up the value(s):

#	Field	LookupField
1	Prod_Key	ProductKey

Specify the fields to retrieve :

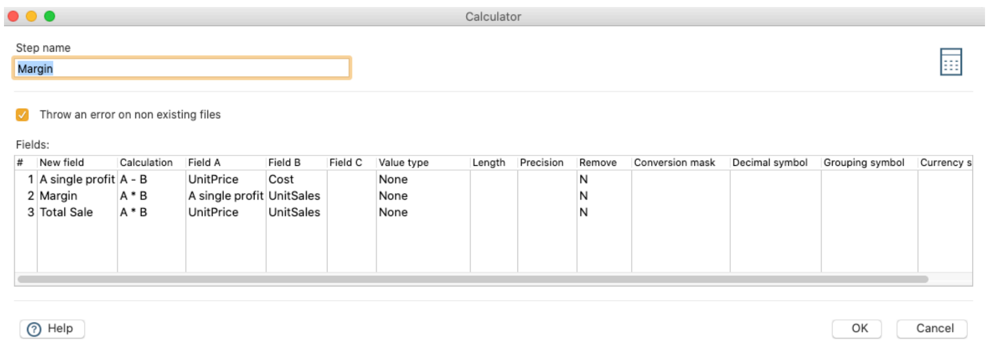
#	Field	New name	Default	Type
1	Cost			Integer

Preserve memory (costs CPU) ✓

Key and value are exactly one integer field

Use sorted list (i.s.o. hashtable)

Figure 16. SaleItem + Cost


Calculator

Step name: Margin

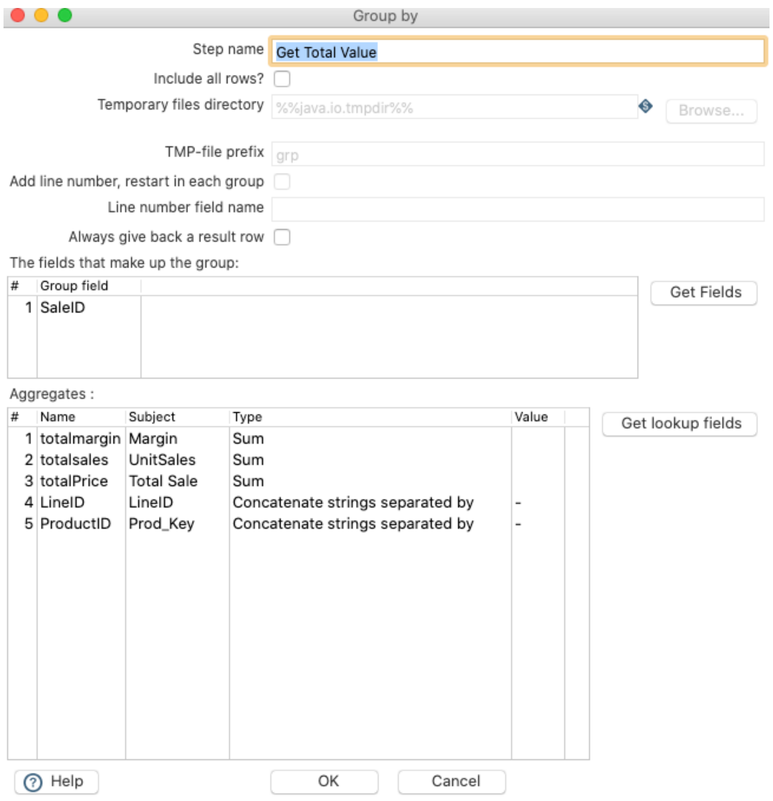
✓ Throw an error on non existing files

Fields:

#	New field	Calculation	Field A	Field B	Field C	Value type	Length	Precision	Remove	Conversion mask	Decimal symbol	Grouping symbol	Currency s
1	A single profit	A - B	UnitPrice	Cost		None			N				
2	Margin	A * B	A single profit	UnitSales		None			N				
3	Total Sale	A * B	UnitPrice	UnitSales		None			N				

Figure 17. Margin

Then UnitSale, UnitPrice, and Margin are combined by the same SaleID in the *SaleItem* table, and the accumulated result values correspond to totalmargin, totalsales, and totalPrice, respectively. At the same time, LineID and Product_Key are combined using a "-" character connection. To preserve the diversity of LineID and ProductID, because a SaleID sometimes corresponds to multiple LineID or ProductID.


Group by

Step name: Get Total Value
Include all rows? ☐
Temporary files directory: %%java.io.tmpdir%%
TMP-file prefix: grp
Add line number, restart in each group ☐
Line number field name:
Always give back a result row ☐

The fields that make up the group:

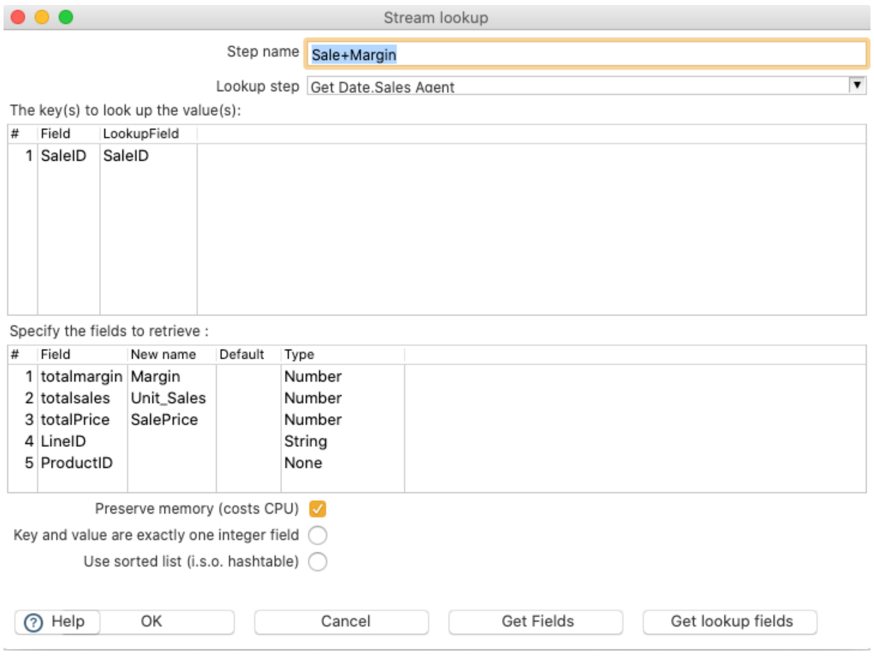
#	Group field
1	SaleID

Aggregates :

#	Name	Subject	Type	Value
1	totalmargin	Margin	Sum	
2	totalsales	UnitSales	Sum	
3	totalPrice	Total Sale	Sum	
4	LineID	LineID	Concatenate strings separated by	-
5	ProductID	Prod_Key	Concatenate strings separated by	-

Figure 18. Get Total Value

The next step is to add to the *Sale* table the totalmargin, totalsales, totalPrice, LineID, and Product_Key corresponding to the SaleID in the above result. And changed to the corresponding attribute name in *Sale FACT* table. Some values are lost here, because the SaleID in the Sale table and the SaleID in the SaleItem table do not correspond one-to-one.


Stream lookup

Step name: Sale+Margin
Lookup step: Get Date.Sales Agent

The key(s) to look up the value(s):

#	Field	LookupField
1	SaleID	SaleID

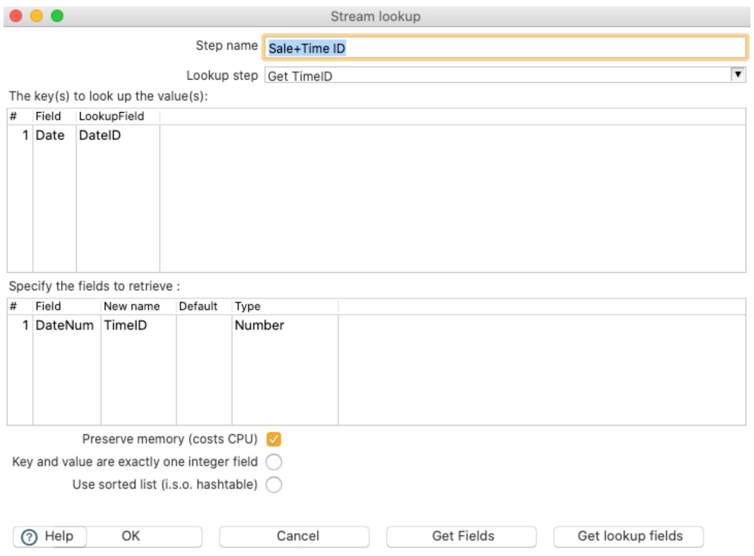
Specify the fields to retrieve :

#	Field	New name	Default	Type
1	totalmargin	Margin		Number
2	totalsales	Unit_Sales		Number
3	totalPrice	SalePrice		Number
4	LineID			String
5	ProductID			None

Preserve memory (costs CPU) ✓
Key and value are exactly one integer field ○
Use sorted list (i.s.o. hashtable) ○

Figure 19. Sale+Margin

Since the Date in the *Sale* table is not the primary key in the *DimDates* table. Therefore, to obtain the primary key, the corresponding DateNum must be queried in the *DimDates* table through Date. The null values generated here will be deleted because some Date displays incorrect values in Sale, such as 29/02 / 2017.



Step name: Sale+Time ID

Lookup step: Get TimeID

The key(s) to look up the value(s):

#	Field	LookupField
1	Date	DateID

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	DateNum	TimeID		Number

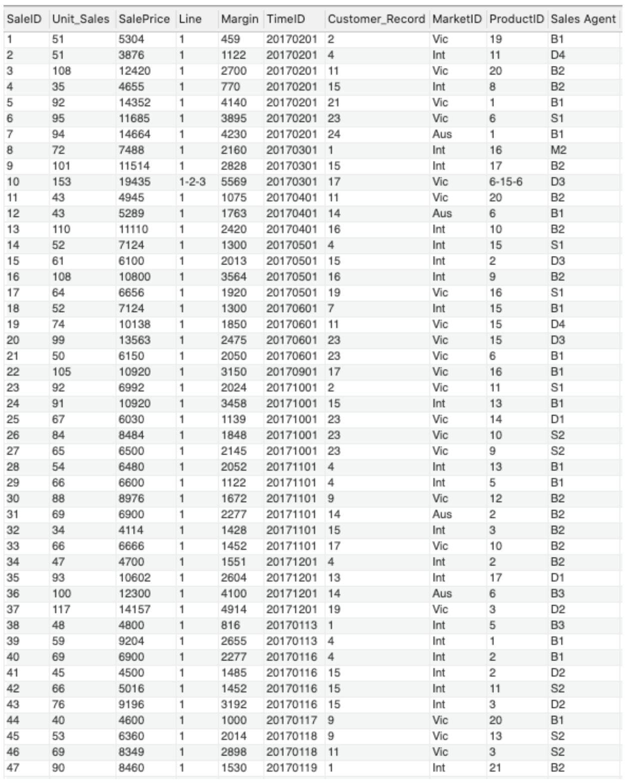
Preserve memory (costs CPU) ✓

Key and value are exactly one integer field

Use sorted list (i.s.o. hashtable)

Figure 20. Sale+Time ID

Finally, sort the required fields and output the final table.



SaleID	Unit_Sales	SalePrice	Line	Margin	TimeID	Customer_Record	MarketID	ProductID	Sales Agent
1	51	5304	1	459	20170201	2	Vic	19	B1
2	51	3876	1	1122	20170201	4	Int	11	D4
3	108	12420	1	2700	20170201	11	Vic	20	B2
4	35	4655	1	770	20170201	15	Int	8	B2
5	92	14352	1	4140	20170201	21	Vic	1	B1
6	95	11685	1	3895	20170201	23	Vic	6	S1
7	94	14664	1	4230	20170201	24	Aus	1	B1
8	72	7488	1	2160	20170301	1	Int	16	M2
9	101	11514	1	2828	20170301	15	Int	17	B2
10	153	19435	1-2-3	5569	20170301	17	Vic	6-15-6	D3
11	43	4945	1	1075	20170401	11	Vic	20	B2
12	43	5289	1	1763	20170401	14	Aus	6	B1
13	110	11110	1	2420	20170401	16	Int	10	B2
14	52	7124	1	1300	20170501	4	Int	15	S1
15	61	6100	1	2013	20170501	15	Int	2	D3
16	108	10800	1	3564	20170501	16	Int	9	B2
17	64	6656	1	1920	20170501	19	Vic	16	S1
18	52	7124	1	1300	20170601	7	Int	15	B1
19	74	10138	1	1850	20170601	11	Vic	15	D4
20	99	13563	1	2475	20170601	23	Vic	15	D3
21	50	6150	1	2050	20170601	23	Vic	6	B1
22	105	10920	1	3150	20170901	17	Vic	16	B1
23	92	6992	1	2024	20171001	2	Vic	11	S1
24	91	10920	1	3458	20171001	15	Int	13	B1
25	67	6030	1	1139	20171001	23	Vic	14	D1
26	84	8484	1	1848	20171001	23	Vic	10	S2
27	65	6500	1	2145	20171001	23	Vic	9	S2
28	54	6480	1	2052	20171101	4	Int	13	B1
29	66	6600	1	1122	20171101	4	Int	5	B1
30	88	8976	1	1672	20171101	9	Vic	12	B2
31	69	6900	1	2277	20171101	14	Aus	2	B2
32	34	4114	1	1428	20171101	15	Int	3	B2
33	66	6666	1	1452	20171101	17	Vic	10	B2
34	47	4700	1	1551	20171201	4	Int	2	B2
35	93	10602	1	2604	20171201	13	Int	17	D1
36	100	12300	1	4100	20171201	14	Aus	6	B3
37	117	14157	1	4914	20171201	19	Vic	3	D2
38	48	4800	1	816	20170113	1	Int	5	B3
39	59	9204	1	2655	20170113	4	Int	1	B1
40	69	6900	1	2277	20170116	4	Int	2	B1
41	45	4500	1	1485	20170116	15	Int	2	D2
42	66	5016	1	1452	20170116	15	Int	11	S2
43	76	9196	1	3192	20170116	15	Int	3	D2
44	40	4600	1	1000	20170117	9	Vic	20	B1
45	53	6360	1	2014	20170118	9	Vic	13	S2
46	69	8349	1	2898	20170118	11	Vic	3	S2
47	90	8460	1	1530	20170119	1	Int	21	B2

Figure 21. The output of Sale Fact table

2.7 Agent_commission Fact Table Transformation

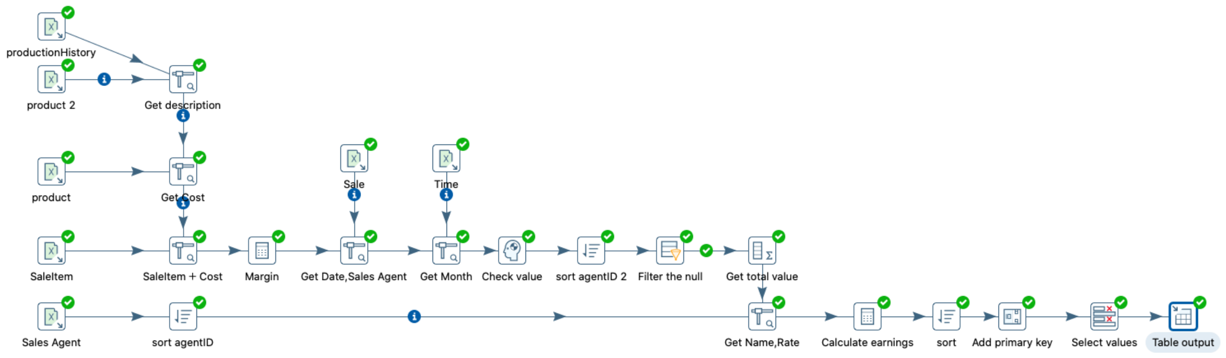
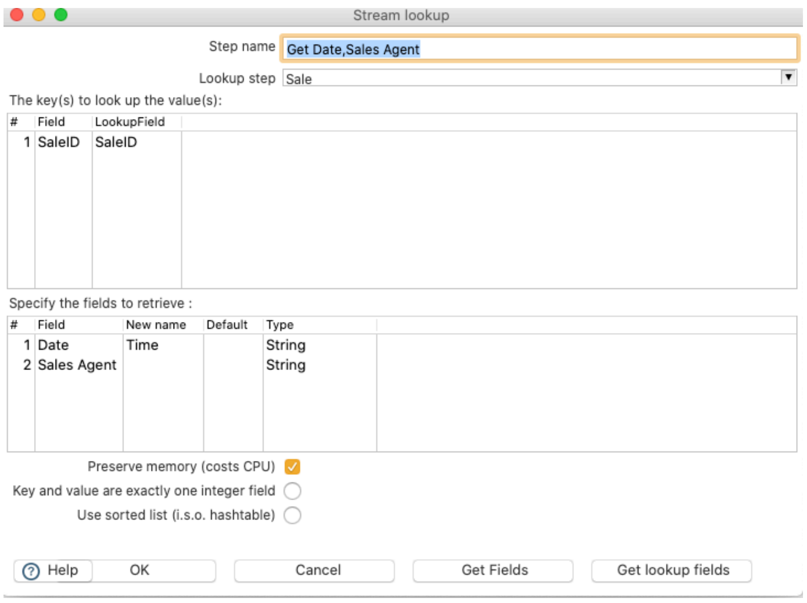


Figure 22. Agent_commission Fact Table Transformation

In this transformation, based on the *SalesItem* table provided by SalesSystem, it provides the required attributes in the *Agent_commision Fact* table through its relationship with other tables.

Adding the Margin corresponding to each SaleID to the *SaleItem* table is the same as that in the Sales Fact Table Transformation.

After *SaleItem* table adds Margin, you need to add the corresponding Date and Sales Agent to connect the *DimeDates* table with the *Sales Agent* table. Query the *Sale table* by SaleID to get the corresponding value.


Stream lookup

Step name: Get Date,Sales Agent

Lookup step: Sale

The key(s) to look up the value(s):

#	Field	LookupField
1	SaleID	SaleID

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	Date	Time		String
2	Sales Agent			String

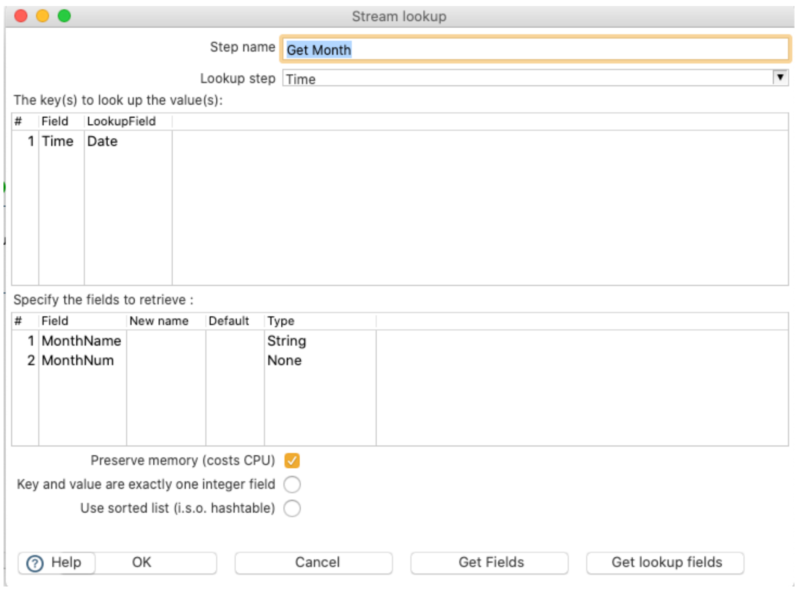
Preserve memory (costs CPU) ✓

Key and value are exactly one integer field ○

Use sorted list (i.s.o. hashtable) ○

Figure 23. Get Date, Sales Agent

After that, query the values corresponding to MonthNum and MonthName in the *DimeDates* table by the Date value. Because the granularity of the *Agent_commision Fact* table is a month, it needs to be sorted and organized according to each month.


Stream lookup

Step name: Get Month

Lookup step: Time

The key(s) to look up the value(s):

#	Field	LookupField
1	Time	Date

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	MonthName			String
2	MonthNum			None

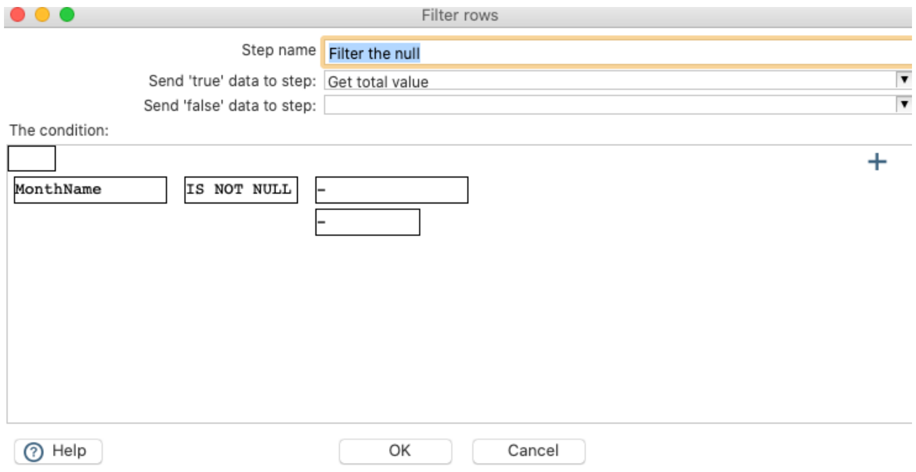
Preserve memory (costs CPU) [✓]

Key and value are exactly one integer field ○

Use sorted list (i.s.o. hashtable) ○

Figure 24. Get Month

There are two problems here. The first is that the date recorded in the *SaleItem* table is wrong. The query cannot find the corresponding month. Another problem is that the SaleID in the *SaleItem* table and the SaleID in the *Sale* table do not correspond one-to-one. Here, all the null-valued rows that are not found can be filtered out.


Filter rows

Step name: Filter the null

Send 'true' data to step: Get total value

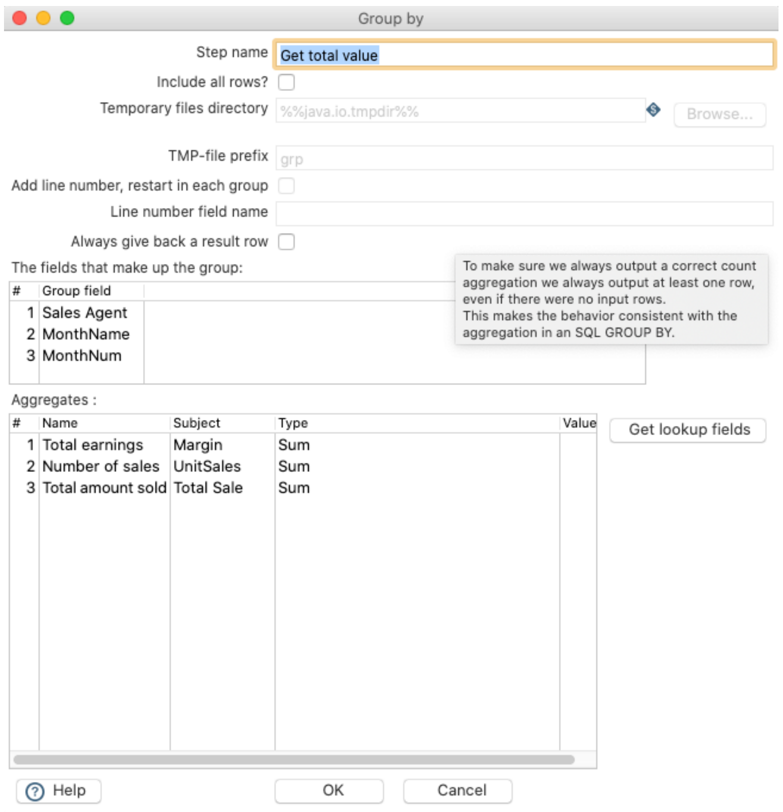
Send 'false' data to step:

The condition:

MonthName IS NOT NULL

Figure 25. Filter the null

Then the values of Margin, UnitSales, and Total Sale are merged and renamed by the same Sales Agent value and the same MonthName value.



Group by

Step name: Get total value

Include all rows? ☐

Temporary files directory: %%java.io.tmpdir%%

TMP-file prefix: grp

Add line number, restart in each group ☐

Line number field name

Always give back a result row ☐

To make sure we always output a correct count aggregation we always output at least one row, even if there were no input rows.
This makes the behavior consistent with the aggregation in an SQL GROUP BY.

The fields that make up the group:

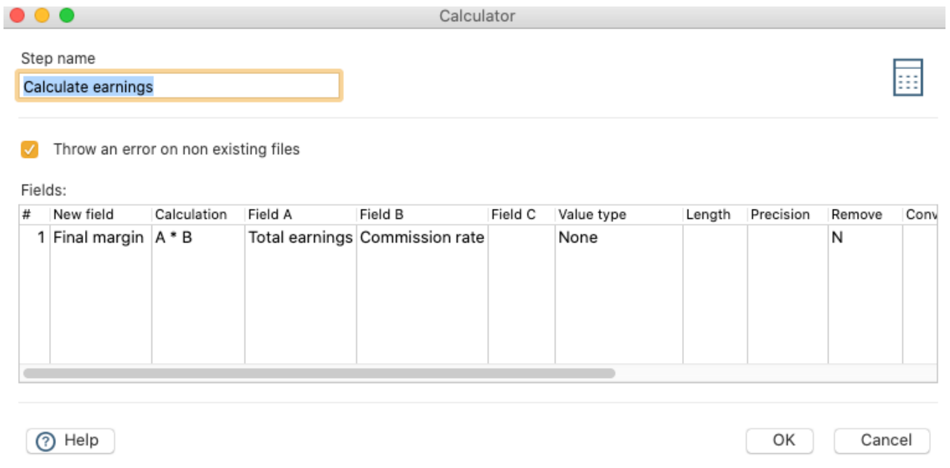
#	Group field
1	Sales Agent
2	MonthName
3	MonthNum

Aggregates :

#	Name	Subject	Type	Value
1	Total earnings	Margin	Sum	
2	Number of sales	UnitSales	Sum	
3	Total amount sold	Total Sale	Sum	

Figure 26. Get total value

Then calculate the real profit through the Commission Rate.



Calculator

Step name: Calculate earnings

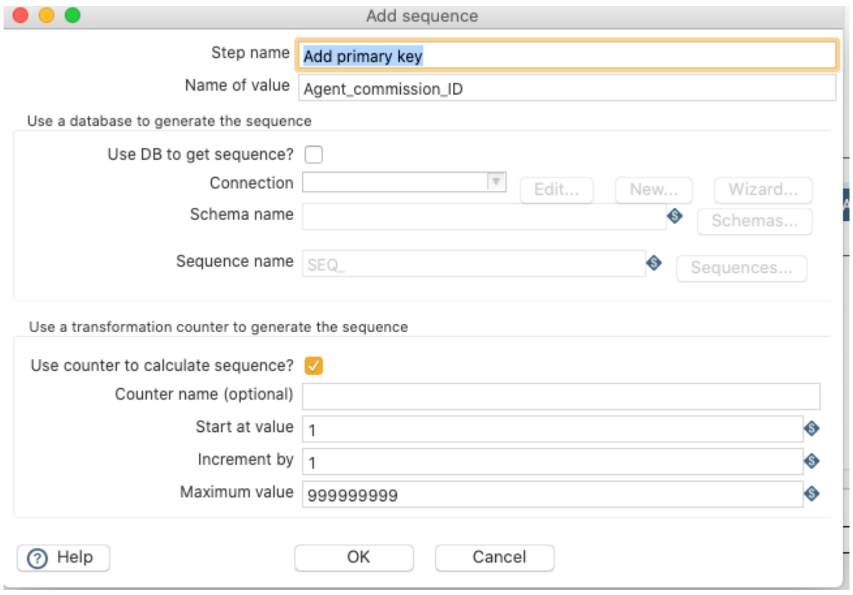
☑ Throw an error on non existing files

Fields:

#	New field	Calculation	Field A	Field B	Field C	Value type	Length	Precision	Remove	Conv
1	Final margin	A * B	Total earnings	Commission rate		None			N	

Figure 27. Calculate earings

Add a primary key for the Agent_Commision Fact table.



Add sequence

Step name: Add primary key

Name of value: Agent_commission_ID

Use a database to generate the sequence

Use DB to get sequence? ☐

Connection:

Schema name:

Sequence name: SEQ_

Use a transformation counter to generate the sequence

Use counter to calculate sequence? ✓

Counter name (optional):

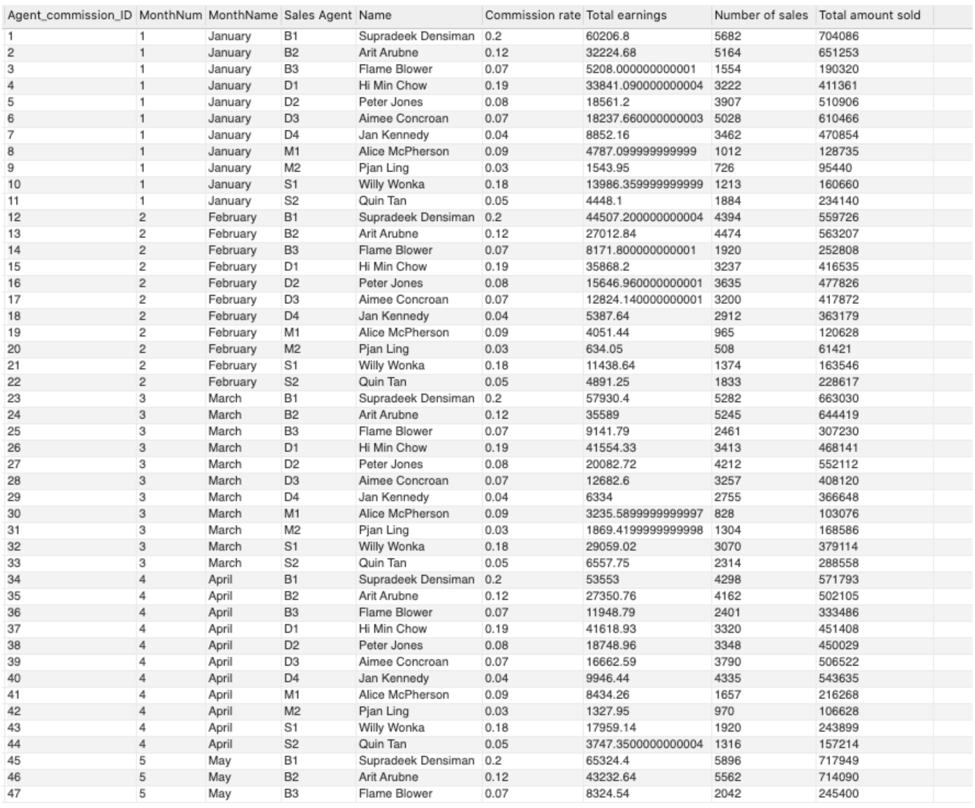
Start at value: 1

Increment by: 1

Maximum value: 999999999

Figure 28. Add primary key

Finally, sort the required fields and output the final table.



Agent_commission_ID	MonthNum	MonthName	Sales Agent	Name	Commission rate	Total earnings	Number of sales	Total amount sold
1	1	January	B1	Supradeek Densiman	0.2	60206.8	5682	704086
2	1	January	B2	Arit Arubne	0.12	32224.68	5164	651253
3	1	January	B3	Flame Blower	0.07	5208.000000000001	1554	190320
4	1	January	D1	Hi Min Chow	0.19	33841.090000000004	3222	411361
5	1	January	D2	Peter Jones	0.08	18561.2	3907	510906
6	1	January	D3	Aimee Concroan	0.07	18237.660000000003	5028	610466
7	1	January	D4	Jan Kennedy	0.04	8852.16	3462	470854
8	1	January	M1	Alice McPherson	0.09	4787.099999999999	1012	128735
9	1	January	M2	Pjan Ling	0.03	1543.95	726	95440
10	1	January	S1	Willy Wonka	0.18	13986.359999999999	1213	160660
11	1	January	S2	Quin Tan	0.05	4448.1	1884	234140
12	2	February	B1	Supradeek Densiman	0.2	44507.200000000004	4394	559726
13	2	February	B2	Arit Arubne	0.12	27012.84	4474	563207
14	2	February	B3	Flame Blower	0.07	8171.800000000001	1920	252808
15	2	February	D1	Hi Min Chow	0.19	35868.2	3237	416535
16	2	February	D2	Peter Jones	0.08	15646.960000000001	3635	477826
17	2	February	D3	Aimee Concroan	0.07	12824.140000000001	3200	417872
18	2	February	D4	Jan Kennedy	0.04	5387.64	2912	363179
19	2	February	M1	Alice McPherson	0.09	4051.44	965	120628
20	2	February	M2	Pjan Ling	0.03	634.05	508	61421
21	2	February	S1	Willy Wonka	0.18	11438.64	1374	163546
22	2	February	S2	Quin Tan	0.05	4891.25	1833	228617
23	3	March	B1	Supradeek Densiman	0.2	57930.4	5282	663030
24	3	March	B2	Arit Arubne	0.12	35589	5245	644419
25	3	March	B3	Flame Blower	0.07	9141.79	2461	307230
26	3	March	D1	Hi Min Chow	0.19	41554.33	3413	468141
27	3	March	D2	Peter Jones	0.08	20082.72	4212	552112
28	3	March	D3	Aimee Concroan	0.07	12682.6	3257	408120
29	3	March	D4	Jan Kennedy	0.04	6334	2755	366648
30	3	March	M1	Alice McPherson	0.09	3235.5899999999997	828	103076
31	3	March	M2	Pjan Ling	0.03	1869.4199999999998	1304	168586
32	3	March	S1	Willy Wonka	0.18	29059.02	3070	379114
33	3	March	S2	Quin Tan	0.05	6557.75	2314	288558
34	4	April	B1	Supradeek Densiman	0.2	53553	4298	571793
35	4	April	B2	Arit Arubne	0.12	27350.76	4162	502105
36	4	April	B3	Flame Blower	0.07	11948.79	2401	333486
37	4	April	D1	Hi Min Chow	0.19	41618.93	3320	451408
38	4	April	D2	Peter Jones	0.08	18748.96	3348	450029
39	4	April	D3	Aimee Concroan	0.07	16662.59	3790	506522
40	4	April	D4	Jan Kennedy	0.04	9946.44	4335	543635
41	4	April	M1	Alice McPherson	0.09	8434.26	1657	216268
42	4	April	M2	Pjan Ling	0.03	1327.95	970	106628
43	4	April	S1	Willy Wonka	0.18	17959.14	1920	243899
44	4	April	S2	Quin Tan	0.05	3747.3500000000004	1316	157214
45	5	May	B1	Supradeek Densiman	0.2	65324.4	5896	717949
46	5	May	B2	Arit Arubne	0.12	43232.64	5562	714090
47	5	May	B3	Flame Blower	0.07	8324.54	2042	245400

Figure 29. The output of Agent_commission Fact Table

2.8 Customer Dimension Table Transformation

The main problem to be solved by Customer Dimension Table Transformation is to combine three Customer tables of different periods into one table. The main problem encountered here is the deletion of duplicate records and the modified way of saving the personal information of the same customer.

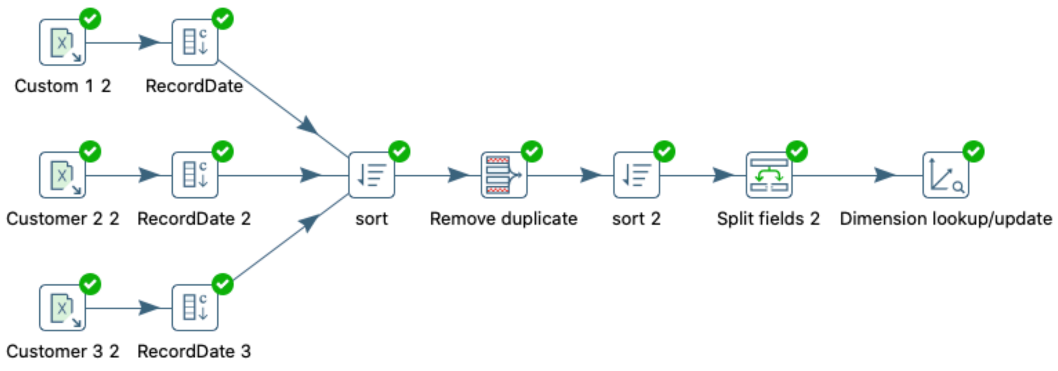
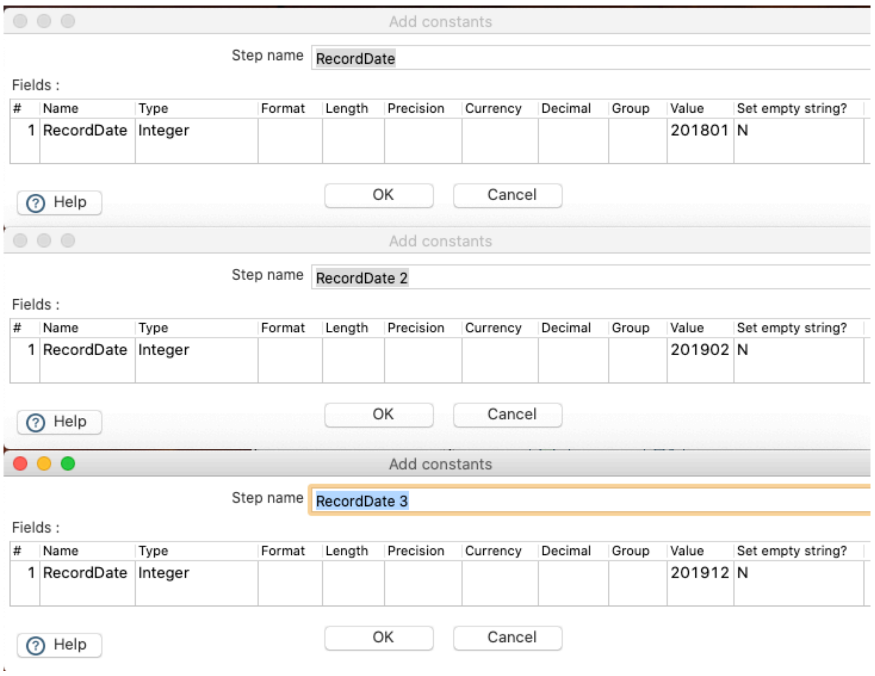


Figure 30. Customer Dimension Table Transformation

First, add the corresponding time fields to the three customer tables.


Add constants

Step name RecordDate

Fields :

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value	Set empty string?
1	RecordDate	Integer							201801	N

Add constants

Step name RecordDate 2

Fields :

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value	Set empty string?
1	RecordDate	Integer							201902	N

Add constants

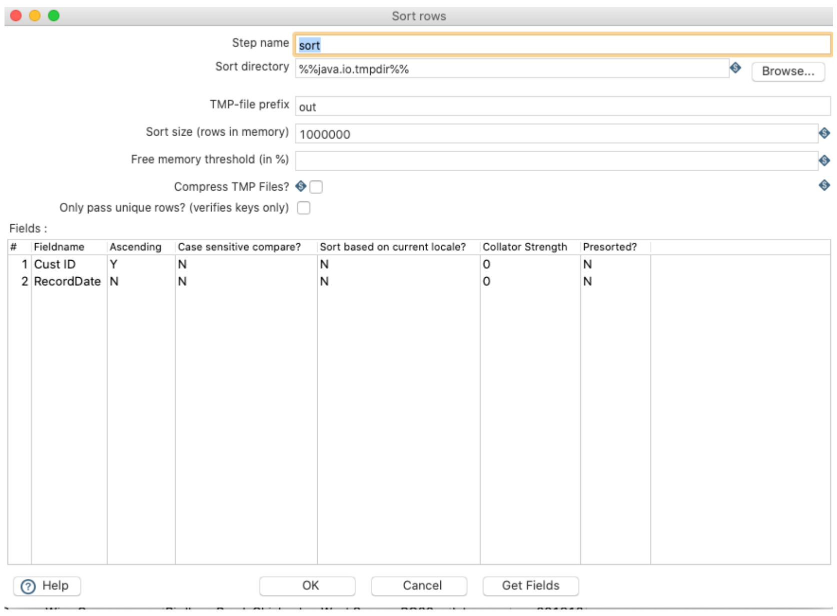
Step name RecordDate 3

Fields :

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value	Set empty string?
1	RecordDate	Integer							201912	N

Figure 31. RecordDate

After that, sort by Cust ID from small to large and RecordDate from large to small, to make each customer display the latest data for the first time.



Step name: sort

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

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

Compress TMP Files? ☐

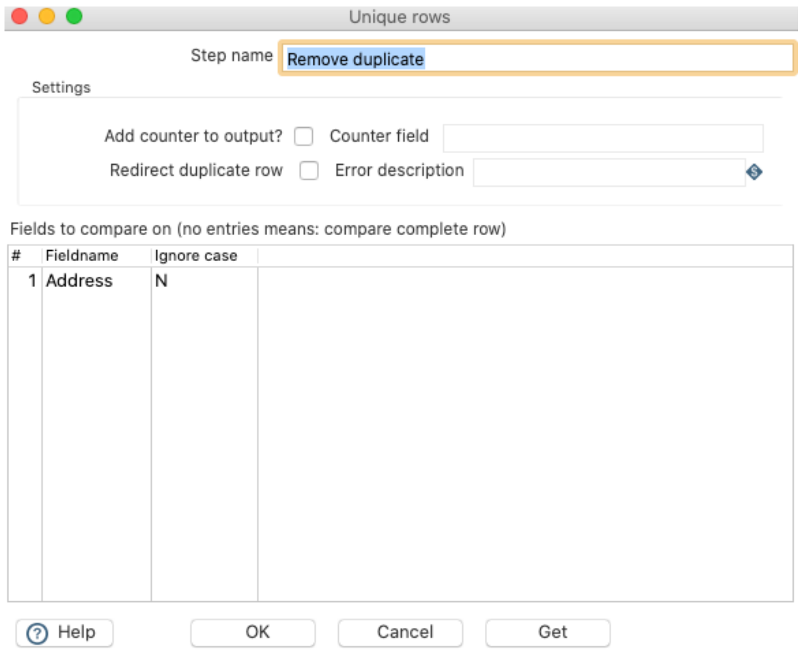
Only pass unique rows? (verifies keys only) ☐

Fields :

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	Cust ID	Y	N	N	0	N
2	RecordDate	N	N	N	0	N

Figure 32. sort

Then remove the duplicate data and sort it again in ascending order according to Cust ID and RecordDate.



Step name: Remove duplicate

Settings

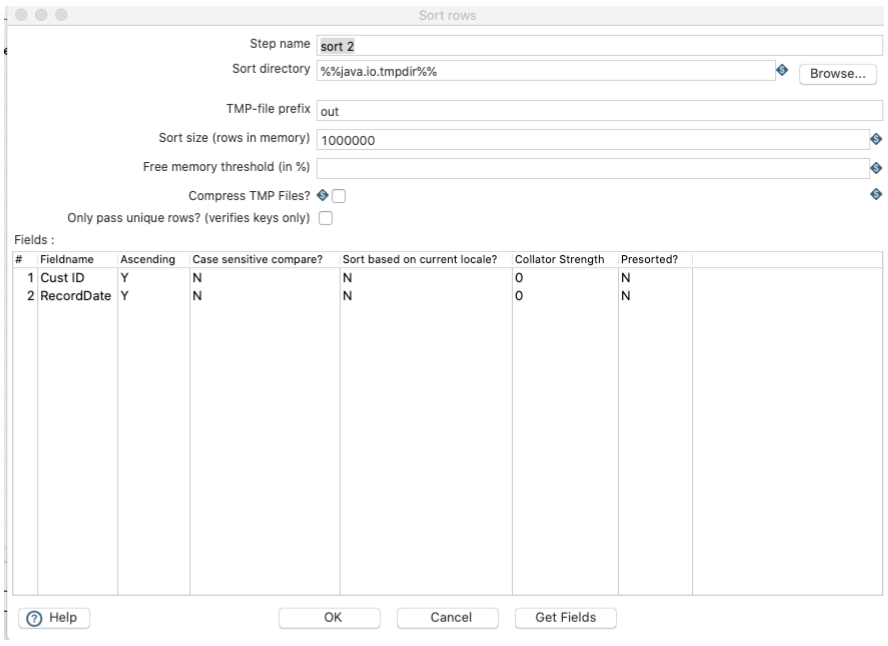
Add counter to output? ☐ Counter field

Redirect duplicate row ☐ Error description

Fields to compare on (no entries means: compare complete row)

#	Fieldname	Ignore case
1	Address	N

Figure 33. Remove duplicate



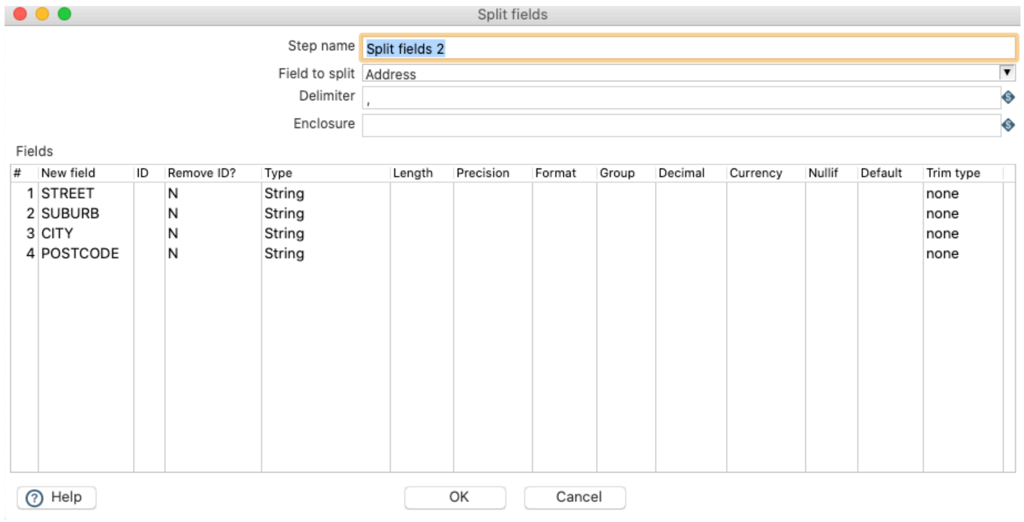
Step name: sort 2
Sort directory: %%java.io.tmpdir%%
TMP-file prefix: out
Sort size (rows in memory): 1000000
Free memory threshold (in %):
Compress TMP Files? ☐
Only pass unique rows? (verifies keys only) ☐

Fields :

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	Cust ID	Y	N	N	0	N
2	RecordDate	Y	N	N	0	N

Figure 34. Sort 2

Then the Address field is divided into STREET, SUBURB, CITY, POSTCODE fields with ',' characters.



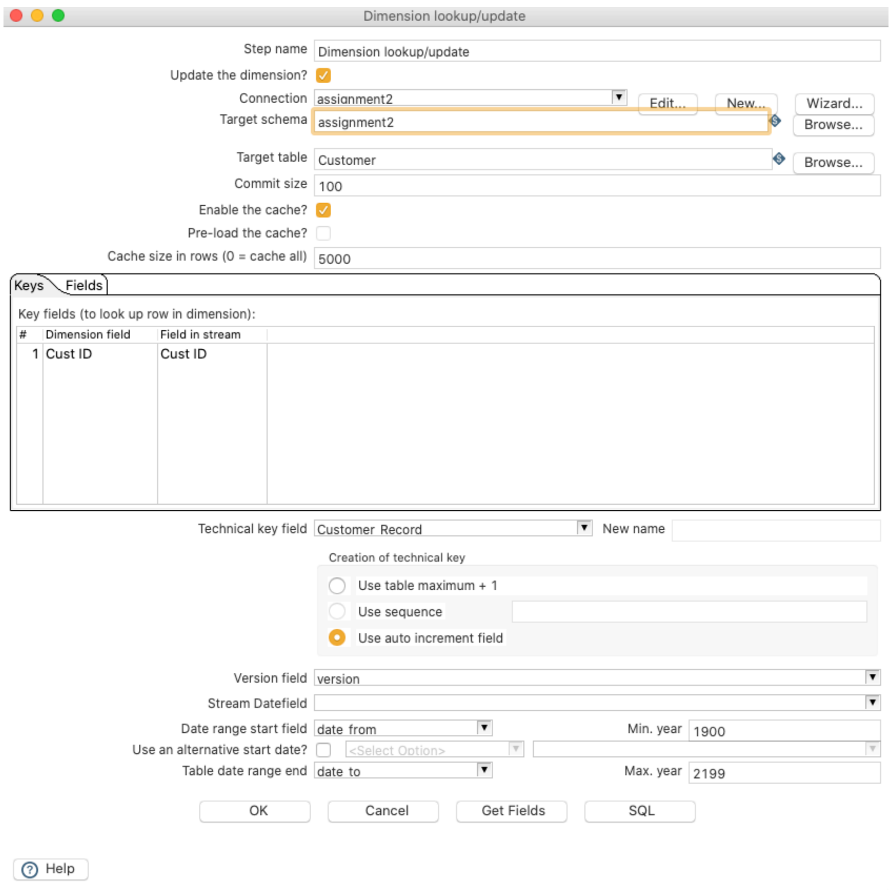
Step name: Split fields 2
Field to split: Address
Delimiter: ,
Enclosure:

Fields

#	New field	ID	Remove ID?	Type	Length	Precision	Format	Group	Decimal	Currency	Nullif	Default	Trim type
1	STREET		N	String									none
2	SUBURB		N	String									none
3	CITY		N	String									none
4	POSTCODE		N	String									none

Figure 35. Split fields

To solve the problem of Slowly changing, SCD Type 2 (keeps the old values by creating a new row for each change) is used to record the data. Therefore, Customer_Record needs to be added as a surrogate key, and the Cust ID field is used as the basis for Screen for changes in other field values to see if there is a change. If there is a change, the value of Version will increase by one.



Dimension lookup/update

Step name: Dimension lookup/update
Update the dimension? ✓
Connection: assignment2
Target schema: assignment2
Target table: Customer
Commit size: 100
Enable the cache? ✓
Pre-load the cache? ☐
Cache size in rows (0 = cache all): 5000

Keys | Fields

Key fields (to look up row in dimension):

#	Dimension field	Field in stream
1	Cust ID	Cust ID

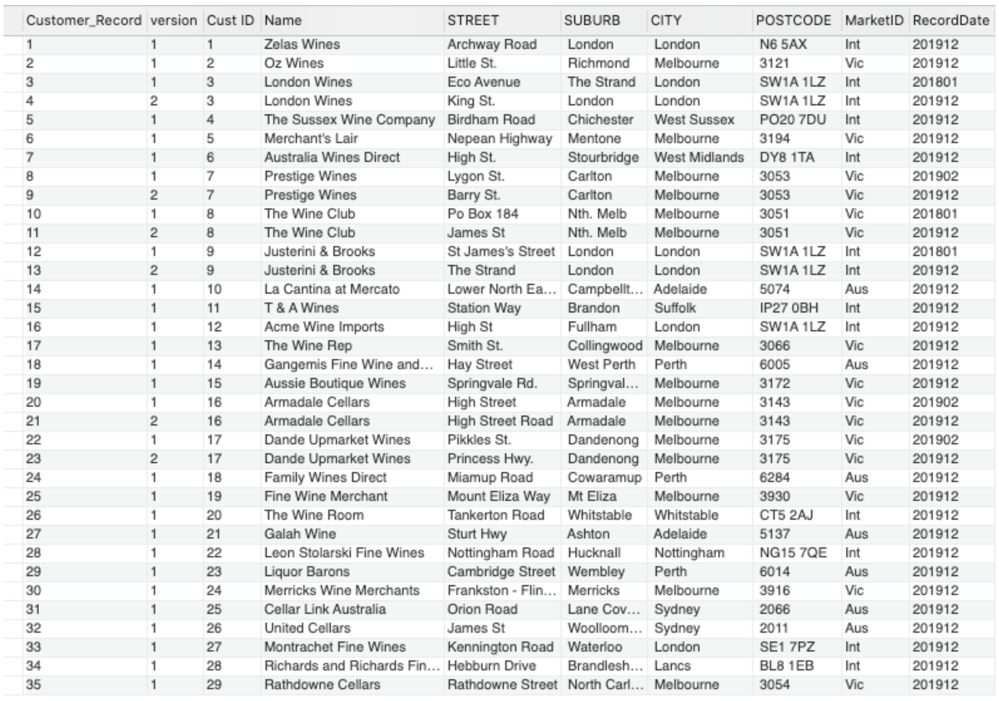
Technical key field: Customer Record
New name:

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

Figure 36. Dimension lookup / update

Finally, sort the required fields and output the final table.



Customer_Record	version	Cust ID	Name	STREET	SUBURB	CITY	POSTCODE	MarketID	RecordDate
1	1	1	Zelas Wines	Archway Road	London	London	N6 5AX	Int	201912
2	1	2	Oz Wines	Little St.	Richmond	Melbourne	3121	Vic	201912
3	1	3	London Wines	Eco Avenue	The Strand	London	SW1A 1LZ	Int	201801
4	2	3	London Wines	King St.	London	London	SW1A 1LZ	Int	201912
5	1	4	The Sussex Wine Company	Birdham Road	Chichester	West Sussex	PO20 7DU	Int	201912
6	1	5	Merchant's Lair	Nepean Highway	Mentone	Melbourne	3194	Vic	201912
7	1	6	Australia Wines Direct	High St.	Stourbridge	West Midlands	DY8 1TA	Int	201912
8	1	7	Prestige Wines	Lygon St.	Carlton	Melbourne	3053	Vic	201902
9	2	7	Prestige Wines	Barry St.	Carlton	Melbourne	3053	Vic	201912
10	1	8	The Wine Club	Po Box 184	Nth. Melb	Melbourne	3051	Vic	201801
11	2	8	The Wine Club	James St	Nth. Melb	Melbourne	3051	Vic	201912
12	1	9	Justerini & Brooks	St James's Street	London	London	SW1A 1LZ	Int	201801
13	2	9	Justerini & Brooks	The Strand	London	London	SW1A 1LZ	Int	201912
14	1	10	La Cantina at Mercato	Lower North Ea...	Campbellt...	Adelaide	5074	Aus	201912
15	1	11	T & A Wines	Station Way	Brandon	Suffolk	IP27 0BH	Int	201912
16	1	12	Acme Wine Imports	High St	Fullham	London	SW1A 1LZ	Int	201912
17	1	13	The Wine Rep	Smith St.	Collingwood	Melbourne	3066	Vic	201912
18	1	14	Gangemis Fine Wine and...	Hay Street	West Perth	Perth	6005	Aus	201912
19	1	15	Aussie Boutique Wines	Springvale Rd.	Springval...	Melbourne	3172	Vic	201912
20	1	16	Armadale Cellars	High Street	Armadale	Melbourne	3143	Vic	201902
21	2	16	Armadale Cellars	High Street Road	Armadale	Melbourne	3143	Vic	201912
22	1	17	Dande Upmarket Wines	Pikkles St.	Dandenong	Melbourne	3175	Vic	201902
23	2	17	Dande Upmarket Wines	Princess Hwy.	Dandenong	Melbourne	3175	Vic	201912
24	1	18	Family Wines Direct	Miamup Road	Cowaramup	Perth	6284	Aus	201912
25	1	19	Fine Wine Merchant	Mount Eliza Way	Mt Eliza	Melbourne	3930	Vic	201912
26	1	20	The Wine Room	Tankerton Road	Whitstable	Whitstable	CT5 2AJ	Int	201912
27	1	21	Galah Wine	Sturt Hwy	Ashton	Adelaide	5137	Aus	201912
28	1	22	Leon Stolarski Fine Wines	Nottingham Road	Hucknall	Nottingham	NG15 7QE	Int	201912
29	1	23	Liquor Barons	Cambridge Street	Wembley	Perth	6014	Aus	201912
30	1	24	Merricks Wine Merchants	Frankston - Flin...	Merricks	Melbourne	3916	Vic	201912
31	1	25	Cellar Link Australia	Orion Road	Lane Cov...	Sydney	2066	Aus	201912
32	1	26	United Cellars	James St	Woolloom...	Sydney	2011	Aus	201912
33	1	27	Montrachet Fine Wines	Kennington Road	Waterloo	London	SE1 7PZ	Int	201912
34	1	28	Richards and Richards Fin...	Hebburn Drive	Brandlesh...	Lancs	BL8 1EB	Int	201912
35	1	29	Rathdowne Cellars	Rathdowne Street	North Carl...	Melbourne	3054	Vic	201912

Figure 37. The output of Customer Dimension Table

3. Design of the data warehouse

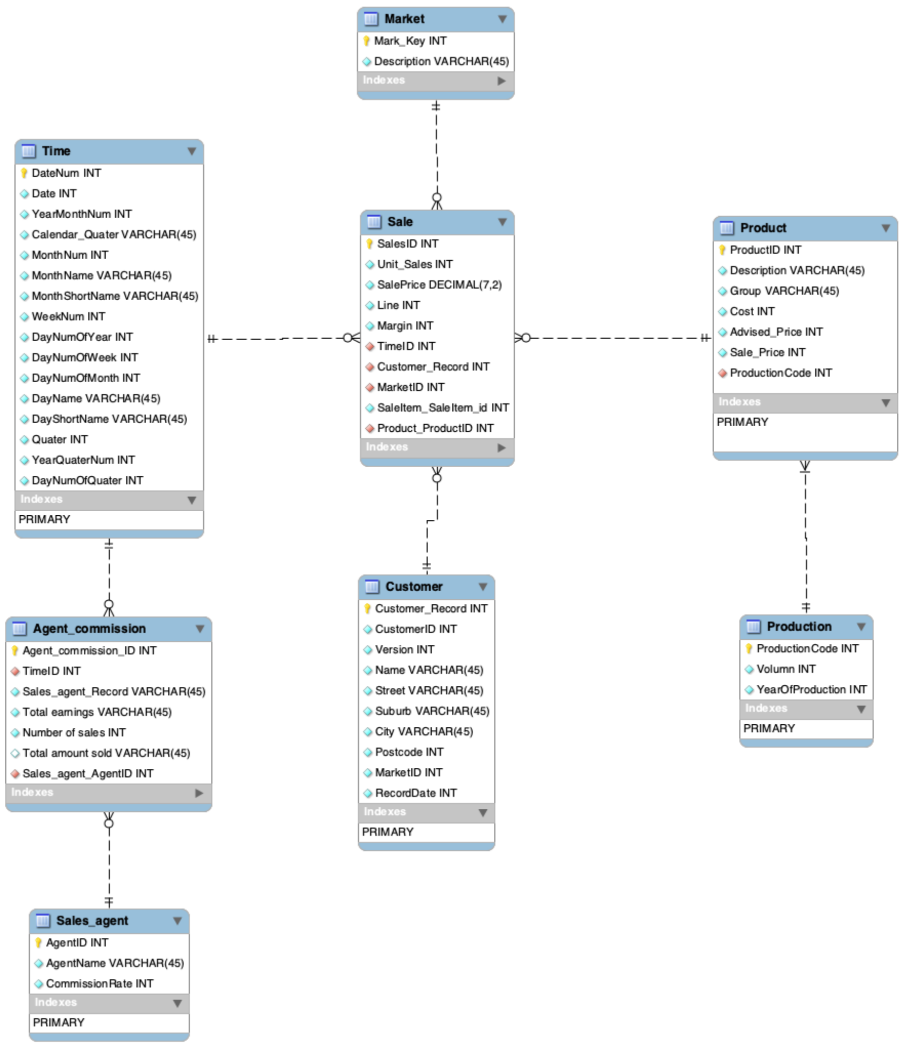


Figure38. Redesign of data warehouse

3.1 Agent_commission Fact table

This fact table is a newly added table. In this table, the primary key is called Agent_commission_ID which is designed for identifying and looking up each agent sales information. The granularity is monthly business performance for each agent. So, this table is mapped with Time dimension table and Sales_agent dimension table and it has primary keys of both these two tables as foreign keys. The measurements of this table are number of sales, total amount sold and total earnings. Therefore, it shows these three measurements of each sale agent for every month.

3.2 Market Dimension table

The market table is simplified from four to two elements. The redundant elements such as the name and region of markets are integrated into descriptions. Mark_key is the primary key of this table to identify the market relative information.

3.3 Time Dimension table

The time table is normalized to a great extent. After redesign, each day, week, quarter and month is represented by number and name. It still indicates both calendar and financial quarters. It offers dates from all different granularity, which is helpful for different kinds of needs.

3.4 Sales_agent Dimension table

Sales_agent table is a dimension table which provides basic agent information for computing the commission of agents. The original agent dimension table is separated into a fact table that is introduced at the beginning of this part and this dimension table, which makes the structure more hierarchical in order to avoid unnecessary confusion.

3.5 Customer Dimension table

In customer dimension table, the address information is subdivided to be more specific and hierarchical, which makes the customer information much easier to look up. Additionally, customers' names are integrated, and the gender and age information are removed due to the redundancy concern.

3.6 Product Dimension table

There are not many modifications in the product dimension table. The only change is to rename some elements about product price to reduce evitable errors. It makes the price of product much clearer especially during the calculations of profits.

4. Data Dictionary

Market dimension table

Name	Market Dimension Table Transformation
Purpose	Analysis sale information from market view
Source Tables/Files	Market (SalesSystem)
Target Tables/Files	Customer dimension table and Sale fact table
Pre Processes	Configurate the source file format *.xlsx* into Pentaho software
Frequency	Monthly

Time dimension table

Name	Time Dimension Table Transformation
Purpose	Offer a view and dimension of time for analysis
Source Tables	DimDates
Target Tables	Sale Fact Table and Agent_commission Fact Table
Pre Processes	Configurate the source file format *xlsx* into Pentaho software
Frequency	None

Product dimension table

Name	Product Dimension Table Transformation
Purpose	Provide the basic product information
Source Tables	Product (SaleSystem), SaleItem (SaleSyetem), Product (ProductionSystem) and ProductionHistory (ProductionSystem)
Target Tables	Production Dimension Table and Sale Fact Table
Pre Processes	Extract UnitPrice for each product from SaleItem, extract ProductionID and cost for each product
Frequency	Monthly

Production dimension table

Name	Production Dimension Table Transformation
Purpose	Offer the basic production information
Source Tables	ProductionHistory(ProductionSystem)
Target Tables	Product Dimension Table
Pre Processes	Configurate the source file format *xlsx* into Pentaho software
Frequency	Yearly

Sales_agent dimension table

Name	Sales_agent Dimension Table Transformation
Purpose	Provide basic agents information
Source Tables	Sales Agent (SalesSystem)
Target Tables	Agent_commission Fact Table
Pre Processes	Configurate the source file format *xlsx* into Pentaho software
Frequency	Monthly

Sales Fact Table

Name	Sales Fact Table Transformation
Purpose	Record the details of each order in the actual transaction
Source Tables	ProductionHistory, Product(ProductionSystem), Product(SalesSystem), SaleItem, Sale, DimDates, Customer
Target Tables	Sales Fact Table
Pre Processes	Query the cost for each Prod_Key, calculate the profit for each SaleID, query the primary key in the timetable for each SaleID, and query the primary key in the customer table for each SaleID, and merge the customer tables
Frequency	Updated daily

Agent_commission Fact Table

Name	**Agent_commission Fact Table Transformation**
Purpose	Record the specific sales situation of each agent every month
Source Tables	ProductionHistory, Product(ProductionSystem), Product(SalesSystem), SaleItem, Sale, DimDates, Sales Agent
Target Tables	Agent_commission Fact Table
Pre Processes	Query the cost for each Prod_Key, calculate the profit for each SaleID, Combine profits and sort by agent and month
Frequency	Updated monthly

Customer Dimension Table

Name	Customer Dimension Table Transformation
Purpose	Analyze the data in Sales Fact Table from the customer's perspective
Source Tables	Customer(SalesSystem)
Target Tables	Customer Dimension Table
Pre Processes	Add the record time of the table, merge the table, deduplicate the data, separate Address into multiple attributes, Dimension lookup / update
Frequency	Updated daily

Word Count: 2277

Appendix – Work Breakdown

Both members participate in the design of ETL process and the redesign of data warehouse.

To be more specific, the details are shown as below:

Name: Chiyu Chen

StudentID: 901265

Contribution:

Design of ETL process

Executive summary

Explanation of part of ETL process

Design of data warehouse

Part of data dictionary

Name: Jie Niu

StudentID: 890649

Contribution:

Design of ETL process

Explanation of part of ETL process

Design of data warehouse

Part of data dictionary