

SPL COOKBOOK

www.raqsoft.com



contents

Chapter 1 Order-related Calculations.....	24
1.1 Access a record with its sequence number.....	25
1.2 Generate a nonexistent group name according to the sequence number in aggregate operation.....	27
1.3 Group records and do calculation by sequence numbers in each group.....	29
1.4 Define subsets by the initial sequence number and the specified step value.....	31
1.5 Loop through sequence numbers to access records and do inter-row calculation.....	33
1.6 Comparison of sequences.....	35
1.7 Alignment calculation between members in sequences.....	37
1.8 Compare whether two sequences are equal.....	39
1.9 Positioning: locate a member in the sequence.....	41
1.10 Positioning: grouping by the positions of members in a sequence.....	43
1.11 Positioning: find a record by the position and do inter-row calculation.....	45
1.12 Positioning: find records by positions and do inter-row calculation.....	48
1.13 Positioning:Group & count by segment.....	50
1.14 Positioning:Group & calculate average value by segment.....	52

1.15 Positioning: obtain records after sorting by their original sequence numbers.....	54
1.16 Positioning: Find positions of members and group members by positions.....	56
1.17 Positioning: check whether a record contains all specified members.....	58
1.18 Positioning: determine whether a record exists by the primary key value.....	60
1.19 Positioning: inter-row calculation over Top N records.....	62
1.20 Select: find the record containing the minimum value.....	64
1.21 Select: find the record containing the maximum value.....	66
1.22 Select: search data by segment	68
1.23 Select:Top N.....	70
1.24 Select: set the primary key to find corresponding records in a related table.....	72
Chapter 2 Complex Query.....	74
2.1 Get records by checking whether a target value is contained in a specified set.....	75
2.2 Get records by checking whether a target value is contained in a specified set (the set is relatively large).....	78
2.3 Get records by matched foreign key values.....	81
2.4 Get records by matched non-foreign-key values.....	84

2.5 Speed up non-foreign-key mapping.....	87
2.6 Get records by matched multi-field foreign key values	90
2.7 An example of self join simplification.....	93
2.8 Get records by mismatched foreign key values.....	96
2.9 Get mismatched records.....	99
2.10 An example of simplifying SQL double negation.....	102
2.11 Get matching records.....	105
2.12 Compare with all results of subquery.....	108
Chapter 3 Top N	111
3.1 Get the maximum value.....	112
3.2 Get the sequence number of the record containing the maximum value and do inter-row calculation.....	115
3.3 Get another field value of the record containing the maximum value.....	118
3.4 Find top N field values	120
3.5 Get the sequence numbers of records containing top N values of a specified field.....	123
3.6 Get records containing top N values in a specified field.....	127

3.7 Get other field values of the records containing top N values of a specified field..... 129

3.8 Get top N records in each group after grouping..... 132

3.9 Perform grouping & aggregation and get top N records in each group..... 135

Chapter 4 Grouping & Aggregation 137

4.1 Aggregation operation: SUM..... 138

4.2 Aggregation operation: MAX & MIN..... 140

4.3 Aggregation operation: AVERAGE..... 142

4.4 Aggregation operation:COUNT..... 144

4.5 Aggregation operation: logic AND 146

4.6 Aggregation operation:logic OR 148

4.7 Aggregation operation: Count distinct members..... 150

4.8 Aggregation operation:MEDIAN 152

4.9 Aggregation operation:RANKING..... 154

4.10 Aggregation operation: An application scenario of RANKING..... 156

Chapter 5 Alignment grouping..... 158

5.1 Group by the specified order, each group keeps only one record.....	159
5.2 Group in specified order.....	161
5.3 Group in specified order and put unmatched records in a new group.....	164
5.4 Group by sequence number, each group keeps only one record.....	167
5.5 Group by sequence number.....	169
5.6 Repeatedly grouped by sequence numbers.....	172
5.7 Group by segments of field values.....	174
5.8 Group by segment according to expression result.....	176
5.9 Group by enumerated conditions, records are not repeatedly grouped.....	178
5.10 Group by enumerated conditions, unmatched records are put in a new group.....	180
5.11 Repeatedly grouped by enumerated conditions.....	182
Chapter 6 Subsets after grouping.....	184
6.1 Inter-row calculation in subsets after grouping.....	185
6.2 Group in the order of record and perform count.....	187
6.3 Ordered conditional grouping	189

6.4 Group by sequence number..... 191

6.5 Multilevel grouping & aggregation..... 193

6.6 Ordered grouping of big data..... 195

6.7 Ordered conditional grouping of big data..... 197

Chapter 7 Loop calculation..... 199

7.1 Merge a sequence and a new member in loop..... 200

7.2 Loop assignment 202

7.3 Loop calculation: complex inter-row calculation..... 204

7.4 Loop calculation: maximum continuous rising days..... 206

7.5 Loop calculation: nested loop..... 208

7.6 Loop calculation: loop number..... 210

7.7 Loop calculation:calculate adjacent data by position during the loop calculation..... 212

7.8 Loop calculation: iterative accumulation..... 214

7.9 Loop calculation:group and calculate ranking..... 216

7.10 Loop calculation: calculate dense ranking in each group..... 218

- 7.11 Loop calculation: iterative sum..... 220
- 7.12 Loop calculation: user-defined iterative calculation..... 222
- Chapter 8 Join query over multiple tables..... 224**
 - 8.1 Perform filtering through multi-level association..... 225
 - 8.2 Switch foreign key field values to the corresponding records..... 227
 - 8.3 Get records by matched foreign key values..... 229
 - 8.4 Get records by mismatched foreign key values..... 231
 - 8.5 Join query over two tables..... 233
 - 8.6 Perform a multi-field join and conditional filtering over two tables..... 235
 - 8.7 Join query over multiple tables..... 237
 - 8.8 Join two tables of the same order by merging..... 239
 - 8.9 Join big data tables of the same order by merging..... 241
 - 8.10 Perform a left join by multi-field primary key of dimension table..... 243
 - 8.11 Perform a left join between two tables..... 245
 - 8.12 Perform a full join between two tables..... 247

8.13 Cartesian product with filter condition.....	249
8.14 Use Cartesian product to calculate matrix multiplication.....	251
8.15 Use left join to calculate Cartesian product.....	253
8.16 Join query between big data tables and large dimension table.....	255
8.17 Fast join query between small data table and large dimension table.....	257
8.18 Fast join query over same-order data tables and large dimension table	259
8.19 Join two tables through locating records by sequence numbers.....	261
8.20 Perform an alignment join by positions to shuffle values of a field.....	263
8.21 Perform alignment join over multiple tables by sequence numbers.....	265
8.22 Cross Apply operation.....	267
8.23 Outer Apply operation.....	269
8.24 Convert Apply operation to Cartesian product.....	271
8.25 Complex uses of Apply operation.....	273
Chapter 9 Inter-set operations.....	275
9.1 Concatenation of two sets.....	276

9.2 Intersection of two sets.....	278
9.3 Union of two sets.....	280
9.4 Difference of two sets.....	282
9.5 XOR operation of two sets.....	284
9.6 Mixed use of concatenation set and difference set.....	286
9.7 Set operations of sequences: intersection and union.....	288
9.8 Concatenation of all set members in a sequence.....	290
9.9 The union of all set members in a sequence.....	292
9.10 Merge same-order sets in the current order to calculate concatenation.....	294
9.11 Merge same-order sets in the current order to calculate union.....	296
9.12 Merge same-order sets in the current order to calculate intersection.....	298
9.13 Merge same-order sets in the current order to calculate XOR.....	300
9.14 Merge same-order sets in the current order to calculate difference.....	302
9.15 Merge table sequences by primary key to calculate concatenation	305
9.16 Merge table sequences to find differences.....	307

9.17 Merge unordered tables to calculate union.....	309
9.18 Aggregation of sequences: union & difference.....	311
9.19 Aggregation of sequences:intersection.....	313
9.20 Perform mixed set operations over two small files.....	315
9.21 Perform complex set operations over two small files.....	317
9.22 Merge two big data tables to calculate concatenation.....	319
9.23 Merge two big data tables to calculate union.....	321
Chapter 10 Transposition.....	323
10.1 Row to column transposition.....	324
10.2 Column to row transposition.....	329
10.3 Bidirectional transposition.....	333
10.4 Dynamic row to column transposition.....	337
10.5 Row to column transposition with dynamic columns by filling into a table.....	339
10.6 Convert multiple rows to multiple rows of another form.....	342
10.7 Transpose rows to columns by position-based value assignment.....	344

10.8 Transpose rows to columns, and do inter-column calculations at the same time.....	346
10.9 Dynamic transposition after the main and sub table join.....	348
10.10 Dynamic row to column transposition after multi-table join.....	350
10.11 Transposition in side by side column group layout.....	353
Chapter 11 Recursion.....	355
11.1 Recursively search single references	356
11.2 Traverse all files in the directory.....	358
11.3 Recursively search all references by loop.....	360
11.4 Recursively search references until the specified value.....	364
11.5 Search the upper level reference.....	368
11.6 Find records with the specified value in the reference chain with the parent value listed.....	371
11.7 Search for leaf records	374
11.8 Find all upper level references.....	377
11.9 Hanoi Tower problem	380
11.10 Pirate treasure division problem.....	383

11.11 Traverse the directories to summarize all the files..... 385

Chapter 12 Using structured text data..... 387

12.1 Filter small files..... 388

12.2 Read certain fields in a text file..... 390

12.3 Read data in a text file using specified separator..... 392

12.4 Aggregate data in a small file to get sum..... 394

12.5 Inter-column calculation in a small file..... 396

12.6 Perform comprehensive calculations using small text files..... 398

12.7 Read untitled structured text data 400

12.8 Read a text file using specified data type and format..... 402

12.9 Read structured text data according to the specified character set..... 404

12.10 Sort data in a small text file in ascending order..... 406

12.11 Sort data in a small text file in descending order..... 408

12.12 Sort structured data in a small text file by multi fields in specified order..... 410

12.13 Perform grouping & aggregation over a small file..... 412

12.14 Perform filter after grouping over a small file.....	414
12.15 Deduplication for a small file.....	416
12.16 Count distinct for small file data.....	418
12.17 Perform grouping & count distinct in each group over a small file.....	420
12.18 Associatively query data over multiple files.....	422
12.19 Join small files to query non-associative field.....	424
12.20 Join small associative files into a wide table.....	426
12.21 Combine data from multiple text files.....	428
12.22 Divide data in a text file into groups and write them to different files.....	430
12.23 Write data in a text file to different files according to judgements	432
Chapter 13 Using structured big text file.....	434
13.1 Filter a big file.....	435
13.2 Perform aggregate sum over a big text file.....	437
13.3 Inter-column calculation in a big text file.....	439
13.4 Perform comprehensive calculations over a big text file.....	441

13.5 Sort a big text file.....	443
13.6 Sort a big text file in descending order.....	445
13.7 Sort a big text file by multiple fields in specified order.....	447
13.8 Find records in a big data table that match data in another big data table.....	449
13.9 Perform grouping & aggregation over a big file, with small result set.....	452
13.10 Perform grouping & aggregation over a big file, with large result set.....	454
13.11 Filter after grouping over a big file.....	456
13.12 Deduplication of big text file.....	458
13.13 Count distinct over a big text file.....	460
13.14 Group & count distinct in each group over a big text file.....	462
13.15 Group a big file by values of a certain field, and query record containing the max value of another field in each group..	464
13.16 Merge & calculate data in multiple big data files.....	466
13.17 The join filter over a large file and a small file.....	468
13.18 Join a large file and a small file into a wide table to query.....	470
13.19 Merge-join two big files.....	472

13.20 Set operations of multiple big text files.....	474
13.21 Divide a big text file into groups and write them to different files.....	476
13.22 Write data in a large text file to different files according to judgements.....	478
13.23 Organize a fixed-structure big text file into structured data.....	480
13.24 Organize a big file with indefinite-line structure into structured data.....	482
13.25 Find the lines containing keyword in all big text files in the specified directory.....	484
13.26 Replace specified text in all text files under the specified directory.....	486
13.27 Count the frequencies of each word in a big text file.....	488
13.28 Count the frequencies of each letter in a big text file.....	490
13.29 Remove duplicate lines from a big text file.....	492
13.30 Remove repeated paragraphs from a big text file.....	494
Chapter 14 Querying text data directly with SQL	496
14.1 Filter.....	497
14.2 Aggregate.....	499
14.3 Inter-column calculation.....	501

14.4 CASE statement.....	503
14.5 Sort.....	505
14.6 TOP-N.....	507
14.7 Group & Aggregate.....	509
14.8 Filter after grouping.....	511
14.9 Select distinct.....	513
14.10 Count distinct.....	515
14.11 Count distinct in each group after grouping.....	517
14.12 Join query over two text files.....	519
14.13 Join query over multiple files.....	521
14.14 Multi-level join query over multiple files.....	523
14.15 Using nested subquery.....	525
14.16 Using common table expression (CTE)	527
14.17 Using command line to execute SQL.....	529
Chapter 15 Using Excel data.....	531

15.1 Read xlsx data in simple format.....	532
15.2 Read xlsx data with complex header.....	534
15.3 Read free format xlsx data.....	536
15.4 Read the crosstab in an xlsx file	539
15.5 Read the main & sub table in xlsx file.....	541
15.6 Read big xlsx file.....	544
15.7 Write a data table to xlsx file.....	546
15.8 Append data table to xlsx file.....	548
15.9 Write data table to different worksheets of an xlsx file.....	550
15.10 Export a large amount of data to xlsx file.....	552
15.11 Sort after join.....	554
15.12 Specify display attributes.....	556
15.13 Fill in the specified cell or area of an xlsx file.....	558
15.14 Export row-style report to xlsx file.....	560
15.15 Export multi-level grouped report to xlsx file.....	563

15.16 Export crosstab report to xlsx file..... 566

15.17 Merge multiple xlsx files of same structure..... 568

15.18 Split an xlsx file and export it to different xlsx files..... 570

Chapter 16 Using JSON and XML data..... 572

16.1 Import single-layer JSON file..... 573

16.2 Import multi-layer JSON file with same-structure detailed data..... 575

16.3 Import multi-layer JSON file with different-structure detailed data..... 582

16.4 Nested aggregation..... 589

16.5 Get field values recursively & merge members of sequences recursively to get SUM..... 591

16.6 Store a JSON file to the database..... 593

16.7 Store a multi-layer JSON file to multiple database tables..... 595

16.8 Output the data table as an XML string with elements only 603

16.9 Import an element-only XML file and organize it according to specified format..... 605

16.10 Import XML file with both elements and attributes..... 607

16.11 Import XML file, perform alignment merging and then filtering..... 609

16.12 Import elements of the specified layer from an XML file with different element structure.....	612
16.13 Import elements of the specified layer from an XML file with different sub-node element structure.....	614
16.14 Join query over XML file and database data.....	617
16.15 Parse XML data in batches.....	620
16.16 Call external Webservice according to parameters and import XML data.....	622
16.17 Get different data from XML file according to parameters.....	624
Chapter 17 Unstructured text handling.....	627
17.1 Multi-line, fixed-structure text structuralization.....	628
17.2 Varied structure text structuralization.....	630
17.3 Parse text with regular expression and organize it into structured data.....	632
17.4 Parse text with regular expression and organize it into structured data (One record corresponds to multiple lines).....	634
17.5 Read in text and perform transposition.....	636
17.6 Complex text structuralization.....	638
17.7 Search all text files in the specified directory to find the lines containing keywords.....	643
17.8 Replace the specified text in all text files in the specified directory.....	645

17.9 Count the frequencies of each English word in a text file..... 647

17.10 Remove duplicate lines from a text file..... 649

17.11 Count the frequencies of each letter in a text file..... 651

17.12 Remove duplicate paragraph from a text file..... 653

Chapter 18 String & datetime handling..... 655

18.1 Concatenate strings in two columns..... 656

18.2 Concatenate string and other type of value..... 658

18.3 Concatenate members in a sequence..... 660

18.4 Add quotation marks to members when concatenating members of a sequence..... 662

18.5 Convert a table sequence to CSV format..... 664

18.6 Split a string into a sequence of characters..... 666

18.7 Split strings into a sequence of words..... 668

18.8 Use tab as a separator to split a string 670

18.9 Use comma as the separator to split a string 672

18.10 Split a string into two segments by specified separator..... 674

18.11 Split a string with regular expression.....	676
18.12 Parse a string into numerical value.....	678
18.13 Parse a percentage string into a numerical value.....	680
18.14 Automatically parse a string into the proper data type.....	682
18.15 Split a string and parse the split members into proper data types.....	684
18.16 Parse string to table sequence.....	686
18.17 Parse the character type field in a table sequence with regular expression.....	688
18.18 Parse indefinite-structure text with regular expression.....	690
18.19 Use code to parse character type fields in a table sequence.....	692
18.20 Modify the filter condition in the SQL statement.....	694
18.21 Translate standard SQL statements into specified database format.....	696
18.22 Parse and analyze HTML file.....	699
18.23 Parse HTML file to get table sequence.....	701
18.24 Calculate the date N days after a certain date.....	703
18.25 Calculate the number of days between two dates.....	705

18.26 Calculate the number of seconds / minutes between two datetimes.....	707
18.27 Calculate the first day and last day of the week.....	709
18.28 Calculate the average daily sales for a quarter.....	711
18.29 Calculate age.....	713
18.30 Calculate the date N months before a certain date.....	715
18.31 Calculate the date after N working days.....	717
18.32 Get a sequence of working days.....	719
18.33 Get a sequence of dates between two dates.....	721
18.34 Divide the period between two dates equally into n segments.....	723



Chapter 1

SPL COOKBOOK

Order-related Calculation

✦ 1.1 Access a record with its sequence number



Get a record from a table according to its sequence number.

Get the trading information of the first trading day and the last trading day in Shanghai Stock Exchange in 2019.

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
2019/12/25	2980.4276	2981.8805	1.9E11
...

✦ 1.1 Access a record with its sequence number



SPL script is as follows, where A() and A.m() are used to get members:

	A	B
1	=file("000001.csv").import@ct()	/Import data file
2	=A1.select(year(Date)==2019).sort(Date)	/Select records of 2019 and sort them by date
3	=A2(1) A2.m(-1)	/Retrieve information of the first and last trading days

A3	Date	Open	Close	Amount
	2019/01/02	2497.8805	2465.291	9.76E10
	2019/12/31	3036.3858	3050.124	2.27E11

✦ 1.2 Generate a nonexistent group name according to the sequence number in aggregate operation



In an alignment aggregate operation, generate a nonexistent group name according to the sequence number. Based on the *employee* table, calculate the average salaries of employees in California, Texas, New York, and Florida; and the average of those in other states in a new group.

ID	NAME	STATE	SALARY
1	Rebecca	California	7000
2	Ashley	New York	11000
3	Rachel	New Mexico	9000
4	Emily	Texas	7000
5	Ashley	Texas	16000
...

✦ 1.2 Generate a nonexistent group name according to the sequence number in aggregate operation



SPL script is as follows, where A.p(-1) is used to obtain the sequence number of the last member:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>employee</i> table
3	[California,Texas,New York,Florida]	/Create state sequence
4	=A2.align@an(A3,STATE)	/Group the <i>employee</i> table in alignment by state; @a option returns all matching members in each group, and @n option puts all mismatched members in a new group at the end.
5	=A4.new(if (#>A3.p(-1),"Other",STATE):STATE,~.avg(SALARY):AvgSalary)	/Calculate the average salary of each group to generate a new table sequence. A.p(-1) gets the sequence number of the last member, then rename the group <i>Other</i> .

A5	STATE	SALARY
	California	7700.0
	Texas	7592.59
	New York	7677.77
	Florida	7145.16
	Other	7308.1

✦ 1.3 Group records and do calculation by sequence numbers in each group



We can group records and do calculation by sequence numbers in each group.
Here is a table that records daily attendance information, as shown below:

Per_Code	in_out	Date	Time	Type
1110263	1	2013-10-11	09:17:14	In
1110263	6	2013-10-11	11:37:00	Break
1110263	5	2013-10-11	11:38:21	Return
1110263	0	2013-10-11	11:43:21	NULL
1110263	6	2013-10-11	13:21:30	Break
1110263	5	2013-10-11	14:25:58	Return
1110263	2	2013-10-11	18:28:55	Out

Every seven pieces of data are one group. We want to convert them into the following result:

Per_Code	Date	In	Out	Break	Return
1110263	2013-10-11	9:17:14	18:28:55	11:37:00	11:38:21
1110263	2013-10-11	9:17:14	18:28:55	13:21:30	14:25:58

✦ 1.3 Group records and do calculation by sequence numbers in each group



Create the target data structure, arrange every seven records according to the required structure and then populate data into the target table.

SPL script is as follows, where A() and A.m() are used to access members with their sequence numbers:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from DailyTime order by Per_Code,Date,Time")	/Query data and sort it by Per_code, Date and Time
3	=A2.group@o((#-1)\7)	/Group by Per_code and Date
4	=create(Per_Code,Date,In,Out,Break,Return)	/Create an empty sequence table to store the final result
5	=A3.(~([1,7,2,3,1,7,5,6]))	/For each group, use A([1,7,2,3,1,7,5,6]) to get the records one by one, which make an ordered all day record.
6	=A5.conj([~.Per_Code,~.Date] ~.(Time).m([1,2,3,4]) ~.Per_Code,~.Date] ~.(Time).m([5,6,7,8]))	/Arrange values in each record into a sequence, and use A.m() access multiple members at a time
7	>A4.record(A6)	/Fill members of each sequence in A4's table sequence

A4	Per_Code	Date	In	Out	Break	Return
	1110263	2013-10-11	09:17:14	18:28:55	11:37:00	11:38:21
	1110263	2013-10-11	09:17:14	18:28:55	13:21:30	14:25:58

✦ 1.4 Get subsets by the initial sequence number and the specified step value



We can get subsets from a sequence by the initial sequence number and the specified step value, and perform set operations.

Find the prime numbers within 100.

✦ 1.4 Get subsets by the initial sequence number and the specified step value



The SPL script is as follows, where `step()` function is used to get members according to the fixed span:

	A	B
1	<code>=to(100)</code>	/Create a sequence of 1 to 100
2	<code>=to(2,10)</code>	/Create a sequence of 2 to 10
3	<code>=A2.(A1.step(~,~*2))</code>	/For each member in A2, calculate the n multiple within 100 (n > 1)
4	<code>=A1.to(2,)\A3.conj()</code>	Remove 1 and all composite numbers within 100 to get all prime numbers within 100. A3.conj() gets the composite numbers within 100.

A4	Member
	2
	3
	5
	7
	11
	...

✦ 1.5 Loop through sequence numbers to access records and do inter-row calculation



We can access multiple records by looping a numerical sequence of sequence numbers and calculate link relative ratio. Below is the closing prices in the last 10 trading days in 2019 at Shanghai Stock Exchange, calculate the growth rate of closing price for each day compared with the previous day.

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
2019/12/25	2980.4276	2981.8805	1.9E11
...

✦ 1.5 Loop through sequence numbers to access records and do inter-row calculation



SPL script is as follows, where the A.p() function is used to return the sequence numbers of the last 10 members:

	A	B
1	=file("000001.csv").import@ct()	/Import data file
2	=A1.select(year(Date)==2019).sort(Date)	/Select records of 2019 and sort them by date
3	=A2.p(to(-10,-1))	/Use p() function to return the sequence numbers of the last 10 members
4	=A3.new(A2(~).Date:Date, string(A2(~).Close/A2(~-1).Close-1, "0.000%"):Increase)	/Recursively calculate the growth rate of the closing price of each trading day compared with the previous trading day

A4	Date	Increase
	2019/12/18	-0.178%
	2019/12/19	0.001%
	2019/12/20	-0.402%
	2019/12/23	-1.404%
	2019/12/24	0.673%

✦ 1.6 Comparison of sequences



Compare two sequences to filter records.

According to the Olympic medal table, find out which Olympic Games China ranked higher than Russia.

Game	Nation	Medal
30	USA	46,29,29
30	China	38,27,23
30	UK	29,17,19
30	Russia	24,26,32
30	Korea	13,8,7
...

✦ 1.6 Comparison of sequences



SPL script is as follows, where the ">" symbol is used to compare sequences by comparing members successively at same positions:

	A	B
1	=file("Olympic.csv").import@cqt()	/Import Olympic Game Rankings
2	=A1.run(Medal=Medal.split@cq())	/Split Medal field into sequence by commas
3	=A2.group(Game)	/Group by game
4	=A3.select(~.select(Nation=="China").Medal>~.select(Nation=="Russia").Medal)	/Use ">" to compare the medal sequences of China and Russia by comparing the number of gold medals, silver medals and bronze medals in order, and select the games where China ranks higher.
5	=A4.(Game)	/List the desired games

A5	Game
	23
	25
	28
	29
	30

Similarly, we can use "<" and "==" to compare sequences.

✦ 1.7 Alignment calculation between members in sequences



The basic arithmetic operations among members at the same positions between sequences.

Calculate the daily relative yield of SZSE 300 Index (399007) to SZSE Component Index (399001) from December 24 to 26, 2019.

Date	Code	Name	Open	Close	Amount
2020/2/18	399001	Shenzhen	11244.7651	11306.4863	3.19E+11
2020/2/17	399001	Shenzhen	10974.9328	11241.4993	3.12E+11
2020/2/14	399001	Shenzhen	10854.4551	10916.3117	2.77E+11
2020/2/13	399001	Shenzhen	10936.5011	10864.3222	2.87E+11
2020/2/12	399001	Shenzhen	10735.0475	10940.7952	2.66E+11
...

✦ 1.7 Alignment calculation between members in sequences



Syntax $A \ ? \ B$ is used to perform the alignment operation "?" over members of two sequences, where $? \in \{+, -, *, /, \%, \backslash\}$. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=["399007","399001"].(A1.query("select * from StockIndex where code=? and date between '2019-12-23' and '2019-12-26'",~))	/Read the data of SZSE 300 Index and SZSE Component Index from December 23 to 26, 2019; the aim of reading data on Dec. 23 is to calculate the increase
3	=A2.(~.calc(to(2,4),Close/Close[-1]))	/Calculate the daily increase from the 24th to the 26th
4	=A3(1)--A3(2)	/Alignment subtraction between the two sequences to get the relative yield

A4	Member
	0.0031349096521252617
	0.0011897141619391371
	-4.4910504685946595E-4

✦ 1.8 Compare whether two sequences are equal



Compare whether the members at same positions in two sequences are all equal.

Below is a part of a file generated by random sampling. Compare whether same IDs are selected by the two random samplings.

ID	Predicted_Y	Original_Y
10	0.012388464367608093	0.0
11	0.01519899123978988	0.0
13	0.0007920238885061248	0.0
19	0.0012656367468159102	0.0
21	0.009460545997473379	0.0
23	0.024176791871681664	0.0
...

✦ 1.8 Compare whether two sequences are equal



SPL script is as follows, where `cmp()` function is used to compare members at same positions in the two sequences :

	A	B
1	<code>=file("p_old.csv").import@ct()</code>	/Read the first output file
2	<code>=file("p_new.csv").import@ct()</code>	/Read the second output file
3	<code>=cmp(A1.(ID),A2.(ID))</code>	/Compare whether the IDs generated in the files are identical (member values are equal and the order is the same)

A3	Member
	0

A result of 0 indicates that the IDs in the two files are exactly the same.

If the order of IDs is expected to be different, use `eq()` function to compare whether the members of the two sequences are the same:

	A	B
3	<code>=A1.(ID).eq(A2.(ID))</code>	/Compare their ID values only when the order is not necessarily the same

✦ 1.9 Location: locate a member in the sequence



Locate a member's position in the sequence.

In the following *Teachers* table, the first column contains names, the second column contains subjects, followed by the course code (null is empty).

Teachers.txt														
Petitti	Matematica	mif	mig	vif	vig	null	null	null	null	null	null	null	null	...
Canales	Apesca	luc	lud	mac	mad	mic	mid	juc	jud	null	null	null	null	...
Lucero	NavegacionI	lub	luc	lud	lue	mab	mac	mad	mae	mib	mic	mid	mie	...
Bergamaschi	TecPesc	lua	luf	maa	maf	mia	mif	jua	juf	via	vif	null	null	...
...

List available teachers for each course according to the *Teachers* table and the *Courses* table on the right.

Monday	Tuesday	Wednesday	Thursday	Friday
lua	maa	mia	jua	via
lub	mab	mib	jub	vib
luc	mac	mic	juc	vic
lud	mad	mid	jud	vid
lue	mae	mie	jue	vie
luf	maf	mif	juf	vif
lug	mag	mig	jug	vig

✦ 1.9 Location: locate a member in the sequence



SPL script is as follows, where the `pos()` function is used to obtain a member's position.

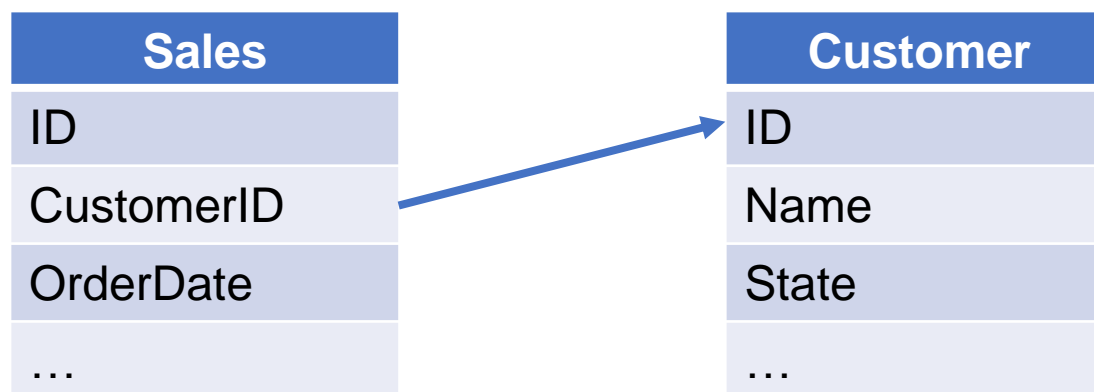
	A	B
1	<code>=file("Teachers.txt").import()</code>	/Import data file
2	<code>=A1.new(#1:professor,~.array().to(3,).select(~!=null):codeArray)</code>	/Generate a two-column table sequence, where the first column is the teacher name and the second column is the course list.
3	<code>=file("Courses.txt").import@t().conj(~.array())</code>	/Import the <i>Courses</i> table and merge them into one sequence
4	<code>=A3.(A2.select(codeArray.pos(A3.~)).(professor))</code>	/Loop the course sequence and use <code>pos()</code> function to find the current course in A2's course list and select available teachers for it
5	<code>=create(Monday,Tuesday,Wednesday,Thursday,Friday).record(A4.(~.concat@c()))</code>	/Create a timetable from Monday to Friday and fill in the teachers accordingly

A5	Monday	Tuesday	Wednesday	Thursday	Friday
	Bergamaschi,Puebla	Bergamaschi,Pue...	Bergamaschi,Puebla	Bergamaschi,Pue...	Bergamaschi,Puebla
	Lucero,Puebla,Lu...	Lucero,Mazza,Pu...	Lucero,Puebla,Chi...	Lucero,Mazza,Pe...	Lucero,Puebla,Vel...
	Canales,Lucero,P...	Canales,Lucero,M...	Canales,Lucero,P...	Canales,Lucero,M...	Lucero,Velasco,Lu...

✦ 1.10 Location: grouping by the positions of members in a sequence



Get positions of members using binary search and perform grouping directly by the positions.
Find customers who didn't place an order in 2014 according to *Sales* table and *Customer* table.



✦ 1.10 Location: grouping by the positions of members in a sequence



SPL script is as follows, where A.pos() function is used to locate the positions of members in the sequence:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Sales where year(OrderDate)=2014")	/Query sales records of 2014
3	=A1.query("select * from Customer")	/Query <i>Customer</i> table
4	=A3.(ID).sort()	/List and sort customer IDs
5	=A2.align(A4.len(), A4.pos@b(CustomerID))	/Use pos() function to get customer IDs and group sales records by the IDs. Since the customer IDs are ordered, @b option is used to do a binary search to speed up the location
6	=A3(A5.pos@a(null))	/Use pos@a() to select all customers without orders information (corresponding values are nulls); only the first-found customer will be returned without the @a option

A6	ID	Name	State	...
	ALFKI	CMA-CGM	Texas	...
	CENTC	Nedlloyd	Florida	...

✦ 1.11 Location: find a record by the position and do inter-row calculation



Find the position of the maximum value, get the corresponding record and perform inter-row calculation.
Based on the *Transaction* table below, calculate the growth rate on the day having the highest closing price at SSE Composite Index compared with the previous day.

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
2019/12/25	2980.4276	2981.8805	1.9E11
...

✦ 1.11 Location: find a record by the position and do inter-row calculation



We need to know the sequence number of the record having the highest closing price, and compare it with the price of the previous trading day to get the result.

SPL script is as follows, where `pmax()` function is used to get the sequence number containing the maximum value:

	A	B
1	<code>=file("000001.csv").import@ct()</code>	/Import data file
2	<code>=A1.sort(Date)</code>	/Sort by date
3	<code>=A2.pmax(Close)</code>	/Get the sequence number of the record with the highest closing price
4	<code>=A2.calc(A3,Close/Close[-1]-1)</code>	/Divide the closing price of the current day by that of the previous day to get the growth rate

Similarly, you can use `pmin()` function to get the sequence number of the minimum value:

	A	B
3	<code>=A3.pmin(Close)</code>	/Get the sequence number of the record with the lowest closing price

✦ 1.11 Location: find a record by the position and do inter-row calculation



There are maybe more than one record that contains the maximum value. If you want to return the sequence numbers of all eligible records, just use @a option in the pmax() function:

	A	B
3	=A2.pmax@a(Close)	/Get the sequence numbers of all eligible records with the highest closing price
4	=A2.calc(A3,Close/Close[-1]-1)	/Recursively divide the closing price of the current day by that of the previous day to get the growth rate

If you want to locate a record from back to front, just use the @z option in the pmax() function:

	A	B
3	=A2.pmax@z(Close)	/Get the sequence number of the record with the highest closing price from back to front

✦ 1.12 Location: find records by positions and do inter-row calculation



Locate positions of multiple records according to a specified condition and perform inter-row calculation.
Based on the stock transaction table below, calculate the growth rate of the transaction amount compared with the previous day on the trading days when the closing price rises by more than 3%.

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
2019/12/25	2980.4276	2981.8805	1.9E11
...

✦ 1.12 Location: find records by positions and do inter-row calculation



We need to find the sequence numbers of the records whose closing prices rise by more than 3%, and compare them with the previous trading day to get the result. SPL script is as follows, where pselect() function is used to locate the sequence numbers of members:

	A	B
1	=file("000001.csv").import@ct()	/Import data file
2	=A1.select(year(Date)==2019).sort(Date)	/Select the stock records of 2019
3	=A2.pselect@a(Close/Close[-1]>1.03)	/Get the sequence numbers of records where the closing prices increase by more than 3%; @a option returns the sequence numbers of all eligible records
4	=A3.new(A2(~).Date:Date, A2(~).Amount/A2(~-1).Amount-1:'Amount increase')	/Recursively divide the current trading amount by trading amount of the previous day to calculate growth rate

A3	Member
	161
	187
	211

A4	Date	Amount increase
	2019/02/25	0.758490566037736
	2019/03/29	0.3344827586206895
	2019/05/10	0.3908629441624365

We can see that the trading volume of the three days when the closing price rose by more than 3% was significantly higher than that of the previous day.

✦ 1.13 Location:Group & count by segment



Group records by sequence numbers of segments and count records in each group.

The following is an employee table.

Based on the salary table below, count the employees in three salary ranges respectively (<8000, >=8000 and <12000, >12000).

ID	NAME	BIRTHDAY	SALARY
1	Rebecca	1974-11-20	7000
2	Ashley	1980-07-19	11000
3	Rachel	1970-12-17	9000
4	Emily	1985-03-07	7000
5	Ashley	1975-05-13	16000
...

✦ 1.13 Location:Group & count by segment



SPL script is as follows, where pseg(x) function is used to locate a segment of records:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>EMPLOYEE</i> table
3	[0,8000,12000]	/Define salary ranges
4	=A2.align@a(A3.len(),A3.pseg(SALARY))	/Use pseg() function to get the corresponding salary range
5	=A4.new(A3(#):SALARY,~.count():COUNT)	/Count the employees in each group

A5	
SALARY	COUNT
0	308
8000	153
12000	39

✦ 1.14 Location:Group & calculate average value by segment



Perform grouping by the sequence numbers of segments and calculate average of each group. Calculate average salary for employees who have been with the company less than 10 years, between 10 to 20 years and not less than 20 years respectively, based on the *EMPLOYEE* table.

ID	NAME	HIREDATE	SALARY
1	Rebecca	2005-03-11	7000
2	Ashley	2008-03-16	11000
3	Rachel	2010-12-01	9000
4	Emily	2006-08-15	7000
5	Ashley	2004-07-30	16000
...

✦ 1.14 Location:Group & calculate average value by segment



SPL script is as follows, where the pseg(x,y) function is used to get a segment of records :

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>EMPLOYEE</i> table
3	[0,10,20]	/Define intervals of stay
4	=A2.align@a(A3.len(),A3.pseg(year(now())-~ ,year(HIREDATE)))	/Use pseg() function to get the corresponding interval for each hire date
5	=A4.new(A3(#):EntryYears,~.avg(SALARY):AvgSalary)	/Calculate the average salary of each group

A5	
EntryYears	AvgSalary
0	6807.69
10	7417.78
20	7324.32

✦ 1.15 Location: obtain records by their original sequence numbers after sorting



Get the original sequence numbers of the sorted members.

List the three eldest employees by their hire dates based on the *EMPLOYEE* table.

ID	NAME	BIRTHDAY	HIREDATE
1	Rebecca	1974-11-20	2005-03-11
2	Ashley	1980-07-19	2008-03-16
3	Rachel	1970-12-17	2010-12-01
4	Emily	1985-03-07	2006-08-15
5	Ashley	1975-05-13	2004-07-30
...

✦ 1.15 Location: obtain records by their original sequence numbers after sorting



A.psord() function is used to get the original sequence numbers of a sorted members.

Note that the psort() function does not change the order of members in a sequence.

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE order by HIREDATE")	/Query <i>EMPLOYEE</i> table and sort it by hire date
3	=A2.psord(BIRTHDAY)	/Get the original sequence numbers for sorted employee's birthdays
4	=A2(A3.to(3).sort())	/Select the three oldest employees in the <i>EMPLOYEE</i> table by the top three birthdays

A5	ID	NAME	BIRTHDAY	HIREDATE
	296	Olivia	1968-11-05	2006-11-01
	440	Nicholas	1968-11-24	2008-07-01
	444	Alexis	1968-11-12	2010-12-01

✦ 1.16 Location: Group members by positions repeatedly



Locate positions of members and then group members by their positions repeatedly.
Group records by label and count the frequencies of each label based on *PostRecord* table.

ID	TITLE	Author	Label
1	Easy analysis of Excel	2	Excel,ETL,Import,Export
2	Early commute: Easy to pivot excel	3	Excel,Pivot,Python
3	Initial experience of SPL	1	Basics,Introduction
4	Talking about set and reference	4	Set,Reference,Dispersed,SQL
5	Early commute: Better weapon than Python	4	Python,Contrast,Install
...

✦ 1.16 Location: Group members by positions overlapped



SPL script is as follows, where A.pos() function is used to locate all positions of the same member:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from PostRecord")	/Query <i>PostRecord</i> table
3	=A2.conj(Label.split(",")).id()	/Separate labels by commas and merge them into a sequence of all unique labels
4	=A2.align@ar(A3.len(),A3.pos(Label.split(", ")))	/pos() function locates all positions of a label, and align@r() function groups post records by positions
5	=A4.new(A3(#):Label,~.count():Count).sort@z(Count)	/Count the number of posts per label and sort results in descending order

A5	Label	Count
	SPL	7
	SQL	6
	Basics	5

✦ 1.17 Location: check whether a record contains all specified members



Get records that contains all specified members.

Find countries whose official languages include both Chinese and English, based on the *Language* table.

Country	Language
China	Chinese
UK	English
Singapore	English
Singapore	Malay
Singapore	Chinese
Singapore	Tamil
Malaysia	Malay
...	...

✦ 1.17 Location: check whether a record contains all specified members



SPL script is as follows. A.contain() function is used to determine whether a specified value is member of a sequence :

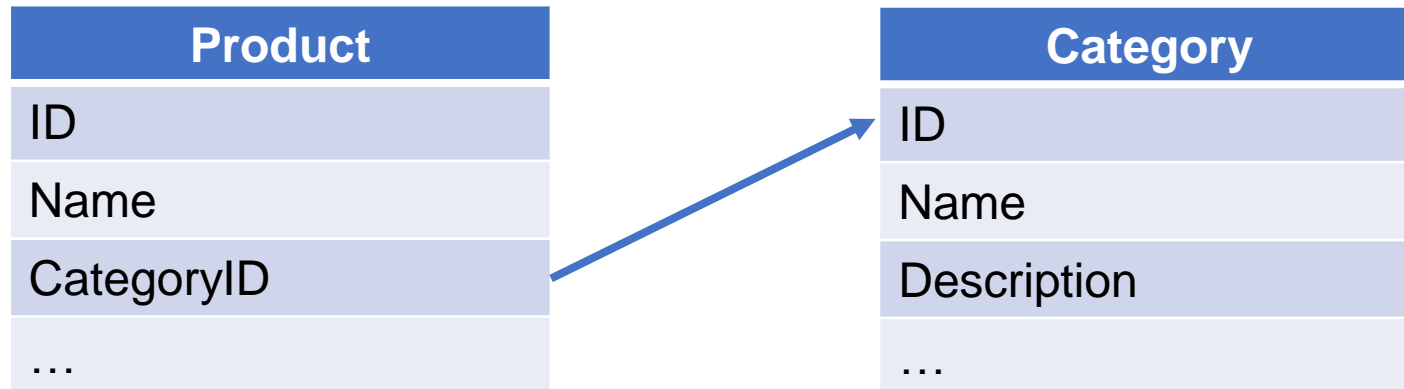
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Language")	/Query <i>Language</i> table
3	=A2.group(Country)	/Group by country
4	=A3.select(~.(Language).contain("Chinese","English"))	/use contain() function to check whether the current country's official languages include both Chinese and English
5	=A4.(Country)	/Get the list of eligible countries

A5	Member
	Singapore

✦ 1.18 Location: determine whether a record exists by the primary key value



Find records in a subtable that don't point to the main table.
The *Product* table and *Category* table are related through CategoryID. Find which product categories are not in the *Category* table.



✦ 1.18 Location: determine whether a record exists by the primary key value



SPL script is as follows, where A.pfind() function is used to get the sequence numbers of primary key values:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Category").keys(ID)	/Query <i>Category</i> table and set ID as the primary key
3	=A1.query("select * from Product")	/Query Product table
4	=A3.select(A2.pfind(CategoryID)=0)	/Use the pfind function to get sequence numbers of category records whose primary key values can't find counterparts in CategoryIDs - a return of 0 indicates that it does not exist among CategoryIDs, and then select product records whose CategoryIDs do not exist in <i>Category</i> table

A4	ID	Name	CategoryID	...
	12	German cheese		...
	26	Spun sugar	9	...

✦ 1.19 Location: inter-row calculation over Top N records



Get the original sequence numbers of Top N records and perform inter-row calculations over the records.
Based on the stock exchange table, calculate the growth rate of transaction amount for each of the three days with the highest closing prices in 2019 at SSE Composite Index, find the increase rate of the transaction amount compared with the previous day.

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
2019/12/25	2980.4276	2981.8805	1.9E11
...

✦ 1.19 Location: inter-row calculation over Top N records



We need to know sequence numbers of the three records with the highest closing prices, and then compare each transaction amount with that of the previous trading day to get the result. SPL is as follows, where ptop() function is used to get the sequence numbers of the three days with the highest closing prices:

	A	B
1	=file("000001.csv").import@ct()	/Import data file
2	=A1.select(year(Date)==2019)	/Select records of 2019
3	=A2.ptop(-3, Close)	/ptop() function gets the sequence numbers of records with the top three closing prices. -3 indicates getting top three in descending order; the positive means getting them in ascending order
4	=A3.run(~=A2(~).Amount/A2(~+1).Amount-1)	/Recursively divide the current trading amount by that of the previous day to calculate the growth rate

A3	VALUE
	154
	156
	157

A4	VALUE
	-0.0278
	-0.0139
	0.0112

✦ 1.20 Select: find the record with the minimum value



Find the record corresponding to the minimum value of the specified field.

According to the *Scores* table, find the student ID with the lowest math score in class one.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 1.20 Select: find the record with the minimum value



SPL is as follows, where minp() function is used to get the record with the minimum value and then the student ID:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Scores where SUBJECT='Math' and CLASS='Class one'")	/Query the math scores of class one
3	=A2.minp(SCORE)	/minp() function gets the record with the lowest score
4	=A3.STUDENTID	/Get the student ID

There may be more than one record with the minimum. To return all records, just use @a option in the function:

	A	B
3	=A2.minp@a(SCORE)	/Get all records with the lowest score
4	=A3.(STUDENTID)	/Return a sequence of student IDs

A3	CLASS	STUDENTID	SUBJECT	SCORE	A4	STUDENTID
	Class one	5	Math	60		5
	Class one	14	Math	60		14

♦ 1.21 Select: find the record with the maximum value



Find the record corresponding to the maximum value.

According to the Olympic medal table, find the nation whose total result holds the first place for the longest time.

Game	Nation	Gold	Silver	Copper
30	USA	46	29	29
30	China	38	27	23
30	UK	29	17	19
30	Russia	24	26	32
30	Korea	13	8	7
...

✦ 1.21 Select: find the record with the maximum value



SPL is as follows, where maxp() function is used to get the record corresponding to the maximum value:

	A	B
1	=file("Olympic.csv").import@cqt()	/Import the Olympic medal table data
2	=A1.sort@z(Game, 1000000*Gold+1000*Silver+Copper)	/Sort by game number and total score in descending order
3	=A2.group@o1(Game)	/Get one for each game, i.e. the first because the data is in order
4	=A3.group@o(Nation)	/Group neighboring countries in the same order
5	=A4.maxp(~.len())	/Get the longest group

A5	Game	Nation	Gold	Silver	Copper
	10	USA	41	32	30
	9	USA	22	18	16
	8	USA	45	27	27
	7	USA	41	27	28

✦ 1.22 Select: search data by segment



Find records by segment according to the specified condition.

According to the *Scores* table, count the number of excellent, pass and fail for English.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 1.22 Select: search data by segment



SPL is as follows, where A.segp() function is used to get members in the sequence corresponding to a segment sequence number:

	A	B
1	=connect("db").query("select * from Scores where SUBJECT='English'")	/Connect to the database and query English scores
2	=create(Assessment,Score).record(["fail",0,"pass",60,"excellent",90])	/Create a base table of assessment and score
3	=A1.derive(A2.segp(Score,SCORE).Assessment:Assessment)	/Use segp function to return assessment corresponding to a score
4	=A3.groups(Assessment;count(1):Count)	/Group and count according to assessment

A4	Assessment	Count
	excellent	6
	fail	4
	pass	18

✦ 1.23 Select:Top N



Get records with the first/last N values of the specified field.

According to the *Scores* table, get the ID of the top two students for each subject in each class.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...



SPL is as follows, where A.top() function is used to get the first / last N members:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Scores")	/Query student scores
3	=A2.group(CLASS,SUBJECT;~.top(-2;SCORE):TOP2)	/Group by class and subject and get the records with the top two scores in each group
4	=A3. conj(TOP2)	/Concatenate the top two records of all classes and subjects

A4	CLASS	STUDENTID	SUBJECT	SCORE
	Class one	4	English	96
	Class one	9	English	93
	Class one	13	Math	97
	Class one	10	Math	97

♦ 1.24 Select: Find a record according to the primary key value



Find a record according to the primary key value.

According to the associated *Course* table and *SelectCourse* table, list the information of courses each student selects, where each course holds a column.

Course		
ID	NAME	TEACHERID
1	Environmental protection and ...	5
2	Mental health of College Students	1
3	Computer language Matlab	8
...

SelectCourse		
ID	STUDENT_NAME	COURSE
1	Rebecca	2,7
2	Ashley	1,8
3	Rachel	2,7,10
...

ID	STUDENT_NAME	COURSE1	COURSE2	COURSE3	...
1	Rebecca	Mental health of College Students	Into Shakespeare		...
2	Ashley	Environmental protection and ...	Modern economics		...
3	Rachel	Mental health of College Students	Into Shakespeare	Music appreciation	...
...

✦ 1.24 Select: Find a record according to the primary key value



SPL is as follows, where find() function is used to find the record containing the current primary key value:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Course").keys(ID)	/Read <i>Course</i> table and set primary key as ID
3	=A1.query("select * from SelectCourse")	/Read <i>SelectCourse</i> table
4	=A3.run(COURSE=COURSE.split@cp())	/Split the courses in the <i>SelectCourse</i> table by commas and assign them to the course field
5	=A4.max(COURSE.len())	/Find the maximum number of selected courses
6	=create(ID,STUDENT_NAME, \${A5.("COURSE"+string(~)).concat@c()})	/Create an empty table having the maximum number of course columns
7	>A4.run(A6.record([ID,STUDENT_NAME] COURSE.(A2. find(~).Name)))	/Loop through <i>SelectCourse</i> table to find, concatenate and insert student IDs, names and the course names obtained through find() function into A6's table sequence

A6	ID	STUDENT_NAME	COURSE1	COURSE2	COURSE3
	1	Rebecca	Mental health of College Students	Into Shakespeare	
	2	Ashley	Environmental protection and ...	Modern economics	
	3	Rachel	Mental health of College Students	Into Shakespeare	Music appreciation

	A	B
7	>A4.run(A6.record([ID,STUDENT_NAME] COURSE.(~.r ow(A2).Name)))	/In A7, you can use row() function to find the records corresponding to the courses since A.find(~) is equivalent to ~.row(A)



Chapter 2

SPL COOKBOOK

Complex Query

✦ 2.1 Get records by checking whether a target value is contained in a specified set



Search for eligible records in a table by checking whether the target field value in a record is included in a specified set.

Based on the employee table, calculate the average salary of each department in the first-tier cities.

ID	NAME	CITY	SALARY
1	Rebecca	Tianjin	7000
2	Ashley	Tianjin	11000
3	Rachel	Shijiazhuang	9000
4	Emily	Shenzhen	7000
5	Ashley	Nanjing	16000
...

✦ 2.1 Get records by checking whether a target value is contained in a specified set



The SQL query:

```
select
    DEPT, avg(SALARY) as SALARY
from
    Employee
where
    STATE in ('Beijing', 'shanghai', 'Guangzhou', 'shenzhen')
group by
    DEPT
```


✦ 2.1 Get records by checking whether a target value is contained in a specified set



When there are no more than 9 constant items in the specified sequence, you can use `A.contain()` function to perform a filter. SPL script is as follows:

	A	B
1	<code>=connect("db").query("select * from Employee")</code>	/Connect to database and query <i>Employee</i> table
2	<code>[Beijing, Shanghai, Guangzhou, Shenzhen]</code>	/Create a constant sequence of first-tier cities
3	<code>=A1.select(A2.contain(CITY))</code>	/Select records with cities included in A2's sequence
4	<code>=A3.groups(DEPT; avg(SALARY):SALARY)</code>	/Group and calculate the average salary of each department

A4	DEPT	SALARY
	Finance	7833.33
	HR	7187.5
	Marketing	7977.27

✦ 2.2 Get records by checking whether a target value is contained in a specified set (the set is relatively large)



Search for eligible records in a table by checking whether the target field value is included in a specified set (here the set is relatively large).

Based on the sales table, calculate the monthly sales of key customers in 2014.

ID	Customer	SellerId	Date	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 2.2 Get records by checking whether a target value is contained in a specified set (the set is relatively large)



The SQL query:

```
select
    month(Date) as Month, sum(Amount) as Amount
from
    Sales
where
    year(Date)=2014
    and Customer in (
        'sAVEA','QUICK','ERNSH','HUN','RATTC','HANAR','FOLKO','QUEEN','MEREP',
        'WHITC','FRANK','KOENE'
    )
group by Month
order by Month
```

✦ 2.2 Get records by checking whether a target value is contained in a specified set (the set is relatively large)



When there are 10 or more constant items, you can first sort the constant sequence and then use @b option with A.contain() function to perform a faster binary search. SPL script is as follows:

	A	B
1	=connect("db").query("select * from Sales")	/Connect to database and query <i>Sales</i> table
2	=["SAVEA","QUICK","ERNSH","HUN","RATTC","HANAR","FOLKO","QUEEN,MEREP","WHITC","FRANK","KOENE"].sort()	/Create a constant sequence of key customers and sort it
3	=A1.select(year(Date)==2014 && A2.contain@b(Customer))	/Select key customer records in 2014
4	=A3.groups(month(Date):Month; sum(Amount):Amount)	/Group and calculate monthly sales

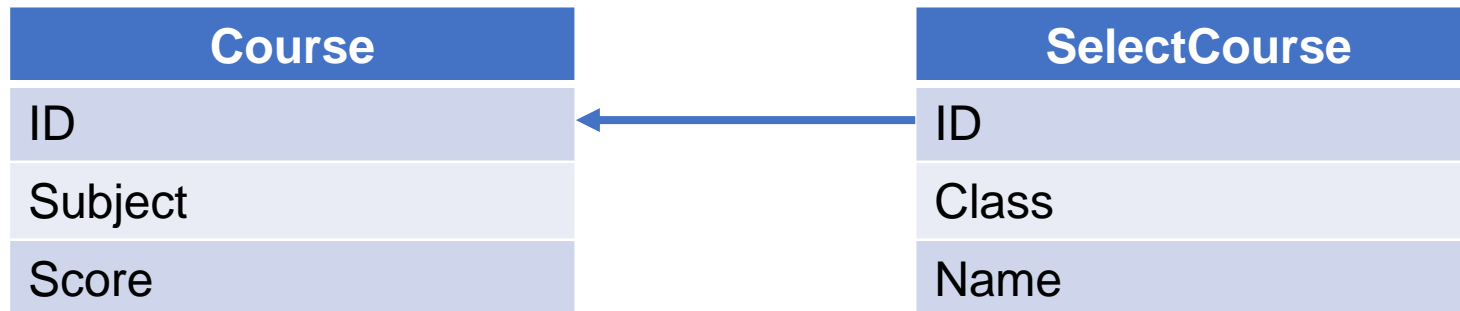
A4	Month	Amount
	1	16947.3
	2	27793.3
	3	14602.7

✦ 2.3 Get records by matched foreign key values



Between two associated tables, search for records according to the foreign key values that can be matched.

According to the *Course* table and *SelectCourse* table, find how many students there are in each class who have selected the "Matlab" course.



✦ 2.3 Get records by matched foreign key values



The SQL query:

```
select
    Class, count(1) as SelectCount
from
    SelectCourse
where
    ID in ( select ID from Course where Name='Matlab' )
group by Class
```

✦ 2.3 Get records by matched foreign key values



@i option is used with the A.join() function to delete the mismatched records. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Course")	/Query <i>Course</i> table
3	=A1.query("select * from SelectCourse")	/Query <i>SelectCourse</i> table
4	=A2.select(Name=="Matlab")	/Select the specified course from the <i>Course</i> table
5	=A3.join@i(ID, A4:ID)	/Use @i option in the join function to perform a join filter
6	=A5.groups(Class; count(1):SelectCount)	/Group and count the students in each class who select the course

A6	Class	SelectCount
	Class 1	3
	Class 2	5

✦ 2.4 Get records by matched non-foreign-key values



Between two associated tables, search for records according to non-foreign-key mapping.

According to the *Score* table and *Student* table, count the students in each class who have at least one subject that is over 80.



✦ 2.4 Get records by matched non-foreign-key values



The SQL query:

```
select
    Class, count(1) as StudentCount
from
    Student
where
    ID in ( select StudentID from Score where Score>80 )
group by Class
```

✦ 2.4 Get records by matched non-foreign-key values



Use a subquery to filter records and then deduplicate them by the joining field. Now it becomes a case of primary key mapping. SPL script is as follows:

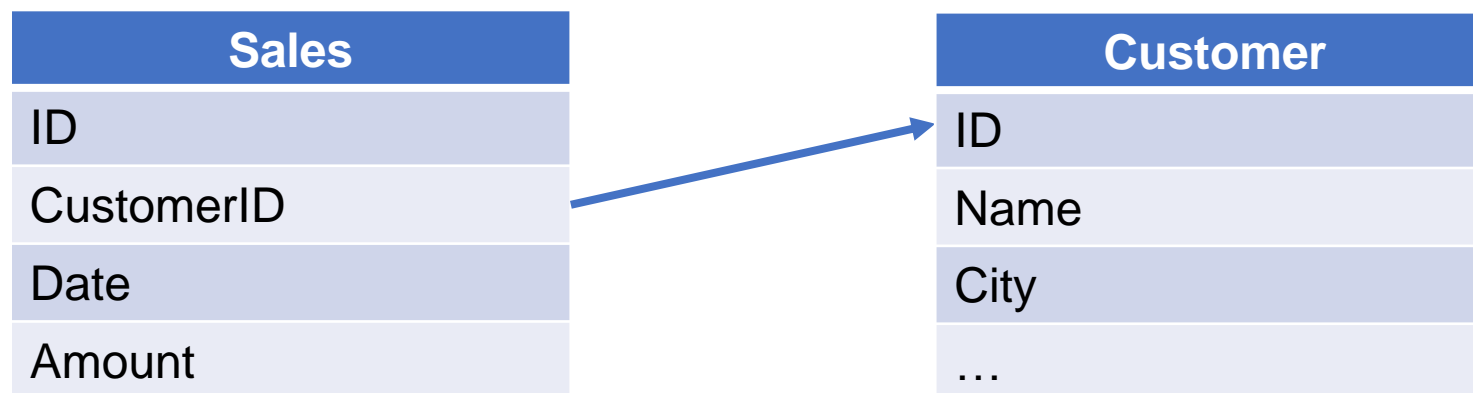
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Student")	/Query <i>Student</i> table
3	=A1.query("select * from Score")	/Query <i>Score</i> table
4	=A3.select(Score>80)	/Select records where the score of at least one subject is over 80
5	=A4.id(StudentID)	/Use id function to deduplicate by Student ID
6	=A2.join@i(ID, A5)	/Use A.join@i() function to perform a join filter
7	=A6.groups(Class; count(1):StudentCount)	/Group and count the number of eligible students in each class

A7	Class	StudentCount
	Class 1	9
	Class 2	11

✦ 2.5 Speed up non-foreign-key mapping



Between two tables, speed up the search for records by non-foreign-key mapping. According to the *Sales* table and *Customer* table, find the number of customers with sales records in 2014 in each city.



✦ 2.5 Speed up non-foreign-key mapping



The SQL query:

```
select
    City, count(1) as CustomerCount
from
    Customer
where
    ID in ( select CustomerID from Sales where year(Date)=2014 )
group by City
```

✦ 2.5 Speed up non-foreign-key mapping



You can use @o option in groups function to speed up the operation if records are ordered by the deduplication field. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Customer")	/Query <i>Customer</i> table
3	=A1.query("select * from Sales where year(Date)=2014 order by CustomerID")	/Query sales records of 2014 and sort them by customer ID
4	=A3.groups@o(CustomerID)	/groups function works with @o option to deduplicate records since they are ordered by the deduplication field
5	=A2.join@i(ID, A4:CustomerID)	/Use A.join@i() function to perform a join filter
6	=A5.groups(City; count(1):CustomerCount)	/Group and count the number of customers in each city

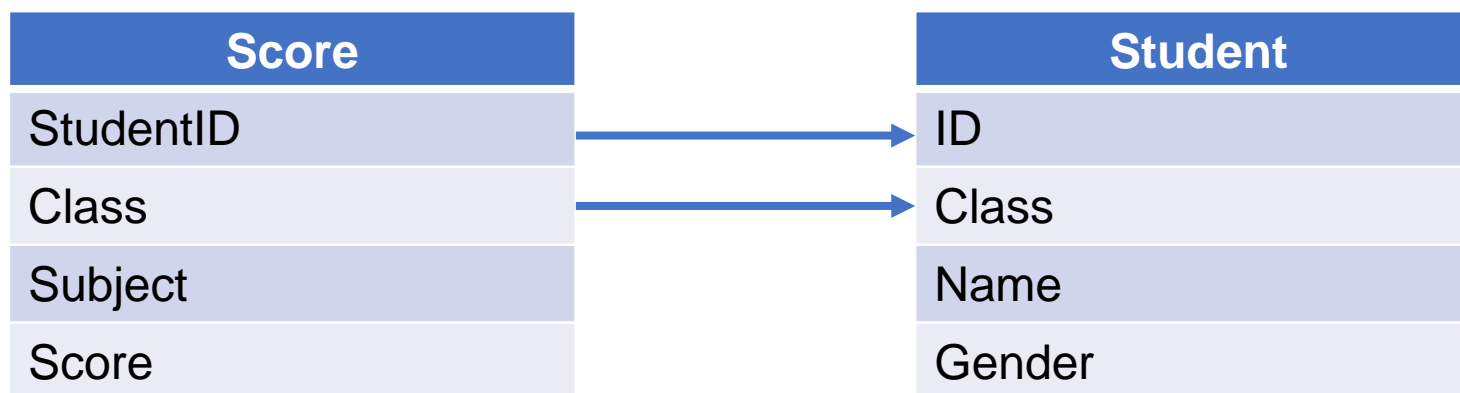
A6	City	CustomerCount
	Dongying	6
	Tangshan	7

✦ 2.6 Get records by matched multi-field foreign key values



Between two associated tables, search for records according to the multi-field foreign key values that can be matched.

Based on the *Score* table and *Student* table, calculate the average score of each boy in class one.



✦ 2.6 Get records by matched multi-field foreign key values



The SQL query:

```
select
    StudentID, avg(Score) as Score
from
    Score
where
    exists (
        select * from Student
        where Class='Class 1' and Gender='Male'
            and Student.Class=Score.Class and Student.ID=Score.StudentID
    )
group by StudentID
```

✦ 2.6 Get records by matched multi-field foreign key values



Our logic is the same as IN subquery. In fact, the EXISTS statement can also be written with the IN statement . SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Score")	/Query the <i>Score</i> table
3	=A1.query("select * from Student")	/Query the <i>Student</i> table
4	=A3.select(Class=="Class 1" && Gender=="Male")	/Select male students in class one
5	=A2.join@i(Class:StudentID, A4:Class:ID)	/Use A.join@i() function to perform a join filter
6	=A5.groups(StudentID; avg(Score):Score)	/Group and calculate the average score of each boy

A6	StudentID	Score
	1	76
	3	74

✦ 2.7 An example of self join simplification



Find desired data through complex self join in a table.

Based on the order table, get the sales amount of orders that spans more than one year's time.

ID	NUMBER	AMOUNT	DELIVERDATE	ARRIVALDATE
10814	1	408.0	2014/01/05	2014/04/18
10814	2	204.0	2014/02/21	2014/04/05
10814	3	102.0	2014/03/14	2014/04/06
10814	4	102.0	2014/04/09	2014/04/27
10814	5	102.0	2014/05/04	2014/07/04
10848	1	873.0	2014/01/06	2014/04/21
...

✦ 2.7 An example of self join simplification



The SQL query:

```
select
    ID, sum(Amount) as Amount
from
    Detail t1
where
    exists (
        select * from Detail t2
        where datediff(t1.ARRIVALDATE, t2.DELIVERDATE)>365
            and t1.ID=t2.ID and t1.NUMBER<>t2.NUMBER
    )
group by ID
```

✦ 2.7 An example of self join simplification



SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Detail")	/Query order details table
3	=A2.group(ID)	/Group by order ID
4	=A3.select(interval(~.min(DELIVERDATE), ~.max(ARRIVALDATE)) > 365)	/Select records of same order that spans more than 365 days
5	=A4.new(ID, ~.sum(AMOUNT):Amount)	/Create a table sequence table and sum the amount of each order

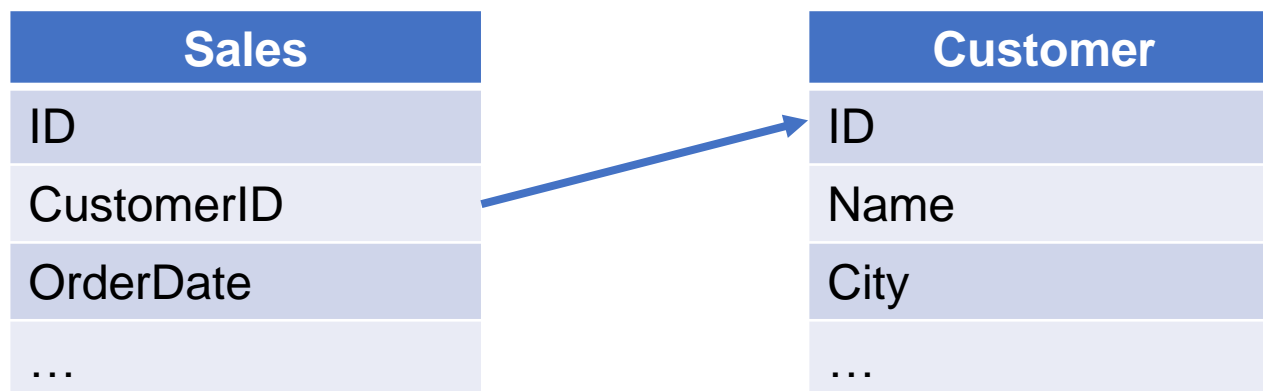
A5	ID	Amount
	10998	6800.0
	11013	4560.0
	11032	20615.0

✦ 2.8 Get records by mismatched foreign key values



Between two associated tables, search for records according to the foreign key values that cannot find matches.

Based on the *Sales* table and *Customer* table, calculate the total sales of each new customer in 2014.



✦ 2.8 Get records by mismatched foreign key values



The SQL query:

```
select
    CustomerID, sum(Amount) as Amount
from
    Sales
where
    CustomerID not in (select ID from Customer)
group by CustomerID
```

✦ 2.8 Get records by mismatched foreign key values



SPL script is as follows, where @d option works with A.join() function to get only the mismatched records:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Sales where year(OrderDate)=2014")	/Query sales records in 2014
3	=A1.query("select * from Customer")	/Query <i>Customer</i> table
4	=A2.join@d(CustomerID ,A3:ID)	/Use A. join@d() to select sales records whose customer IDs do not exist in the <i>Customer</i> table
5	=A4.groups(CustomerID; sum(Amount):Amount)	/Group and calculate the amount of each new customer

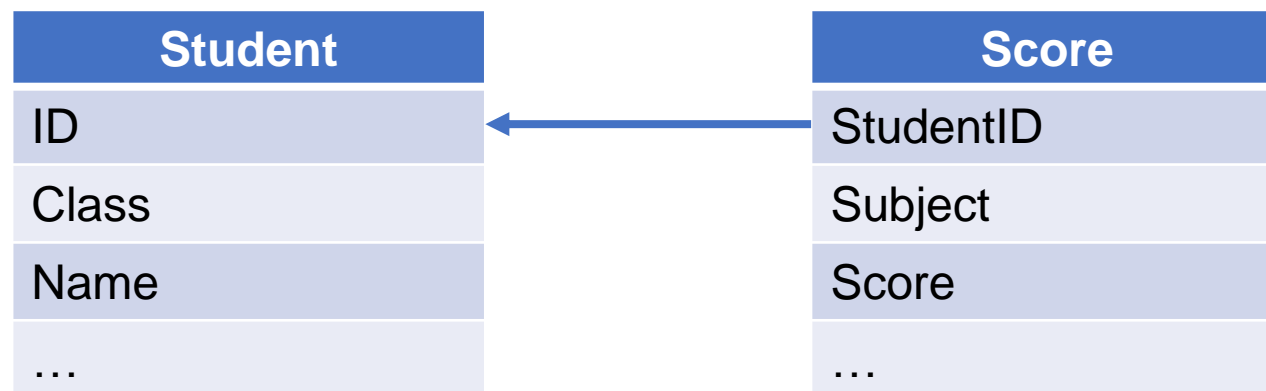
A5	CustomerID	Amount
	DOS	11830.1
	HUN	57317.39

✦ 2.9 Get mismatched records



Between two associated tables, search for records where the joining field values are mismatched.

According to the *Score* table and *Student* table, find students with scores above 80 in all subjects.



✦ 2.9 Get mismatched records



The SQL query:

```
select *  
from Student  
where  
    not exists (  
        select *  
        from Score  
        where  
            Score <= 80 and Score.StudentID = Student. ID  
    )
```


✦ 2.9 Get mismatched records



We only need to find students with no score lower or equal to 80. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Student")	/Query <i>Student</i> table
3	=A1.query("select * from Score")	/Query <i>Score</i> table
4	=A3.select(Score<=80)	/Select records with scores no higher than 80
5	=A4.id(StudentID)	/Deduplicate by Student ID
6	=A2.join@d(ID, A5)	/Use A.join@d() to select mismatched records

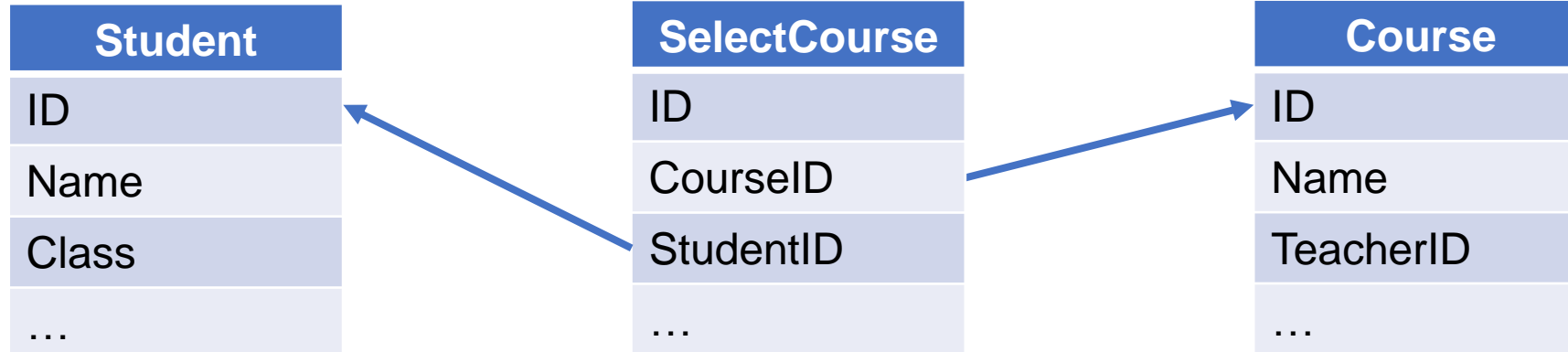
A6	ID	Class	Name
	2	Class 1	Ashley
	16	Class 2	Alexis

✦ 2.10 An example of simplifying SQL double negation



According to a table, find eligible records in another table. The SQL double negation can reduce the amount of computation.

Based on *SelectCourse* table, *Course* table and *Student* table, find the students who select all courses.



✦ 2.10 An example of simplifying SQL double negation



Only need to find students with no course unselected. The SQL query is as follows:

```
Select *  
from Student  
where not exists (  
    select * from Course  
    where not exists (  
        select * from SelectCourse  
        where Course.ID=SelectCourse.CourseID and  
            Student.ID=SelectCourse.StudentID  
    )  
)
```

✦ 2.10 An example of simplifying SQL double negation



SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Student")	/Query <i>Student</i> table
3	=A1.query("select * from Course")	/Query <i>Course</i> table
4	=A1.query("select * from SelectCourse")	Query <i>SelectCourse</i> table
5	=A4.groups(StudentID; icount(CourseID):CourseCount)	/Group <i>SelectCourse</i> table by student ID and count the courses selected by each student
6	=A5.select(CourseCount==A3.len())	/Select the student IDs that all courses are selected
7	=A2.join@i(ID, A6:StudentID)	/Use A.join@i() function to perform a join filter

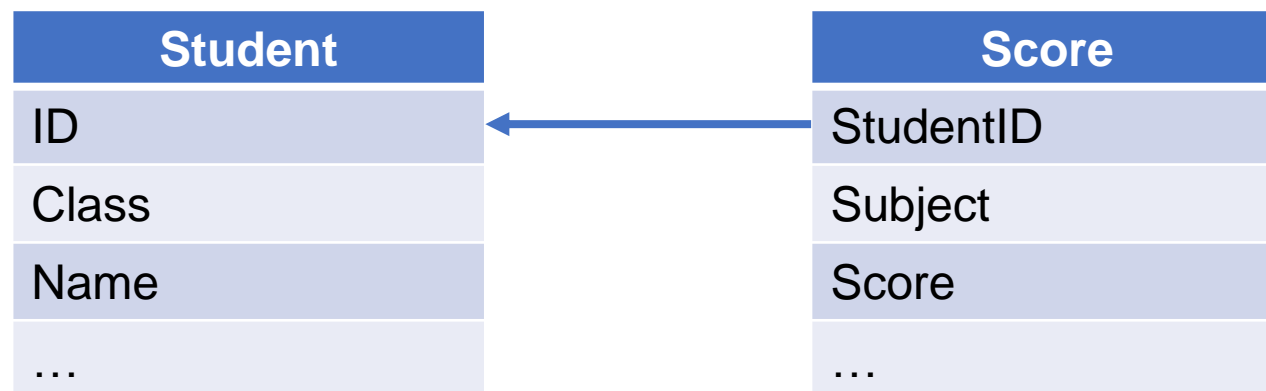
A7	ID	Name	Class
	4	Emily Smith	Class 1

✦ 2.11 Get matching records



Between two associated tables, search for records based on the existence detection of the match.

Based on *Score* table and *Student* table, find students whose score difference between two subjects is more than 30 points.



✦ 2.11 Get matching records



The SQL query:

```
Select * From Student
```

```
Where
```

```
    ID = any (
```

```
        select STUDENTID from Scores t1
```

```
        where
```

```
            SCORE-30 > any (
```

```
                select SCORE from Scores t2
```

```
                where t1.SUBJECT<>t2.SUBJECT and
```

```
                    t1.STUDENTID=t2.STUDENTID
```

```
            )
```

```
    )
```

✦ 2.11 Get matching records



Just compare the highest and lowest scores of each student to see if the difference exceeds 30. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Student")	/Query <i>Student</i> table
3	=A1.query("select * from Score")	/Query <i>Score</i> table
4	=A3.group(StudentID)	/Group <i>Score</i> table by student ID
5	=A4.select(~.max(Score)-~.min(Score)>30)	/Select students where the difference between the highest score and the lowest score is over 30
6	=A5.id(StudentID)	/Deduplicate by Student ID
7	=A2.join@i(ID,A6)	/Use A.join@i() to perform a join filter

A7	ID	Name	Class
	4	Emily Smith	Class 1
	8	Megan	Class 1

✦ 2.12 Compare with all results of subquery



Filter records in a table by comparing with all results of a subquery.

Based on the employee table, find out which employees have higher salaries than all employees in sales department.

ID	NAME	DEPT	SALARY
1	Rebecca	R&D	7000
2	Ashley	Finance	11000
3	Rachel	Sales	9000
4	Emily	HR	7000
5	Ashley	R&D	16000
...

✦ 2.12 Compare with all results of subquery



The SQL query:

```
select
```

```
    *
```

```
from
```

```
    Employee
```

```
where
```

```
    SALARY > all ( select SALARY from Employee where DEPT='sales' )
```

✦ 2.12 Compare with all results of subquery



ALL preceded by a greater than sign is equivalent to max operation; and ALL preceded by a less than sign is equivalent to min operation. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Employee")	/Query <i>Employee</i> table
3	=A2.select(DEPT:"Sales").max(SALARY)	/Select the max salary in sales department
4	=A2.select(SALARY>A3)	/Select employees whose salaries are more than A3

A4	ID	NAME	DEPT	SALARY
	5	Ashley	R&D	16000
	20	Alexis	Administration	16000
	22	Jacob	R&D	18000
	47	Elizabeth	Marketing	17000



Chapter 3

SPL COOKBOOK

Top N

✦ 3.1 Get the maximum value



Calculate the maximum value based on a data table.

Based on the scores table below, find the highest math score in class one.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 3.1 Get the maximum value



SPL script is as follows, where max function is used to get the maximum value:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Scores where SUBJECT='Math' and CLASS='Class one'")	/Query the math scores of class one
3	=A2.max(SCORE)	/Get the highest score

Similarly, you can use the min function to get the minimum value:

	A	B
3	=A2.min(SCORE)	Get the lowest score

✦ 3.1 Get the maximum value



You can also use top to get the highest score, as the following SPL script shows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Scores where SUBJECT='Math' and CLASS='Class one'")	/Query the math scores of class one
3	=A2.top@1(-1, SCORE)	/Get the highest score

When N is ± 1 , top@1(N, x) function returns a numeric value, which is the same with max & min functions.

Value
97



3.2 Get the sequence number of the record with the maximum value and do inter-row calculation



Get the sequence number of the record corresponding to the maximum value of a specified field and perform inter-row calculations.

According to the stock trading table, calculate the growth rate of the day with the highest closing price in 2019, compared with the previous day's closing price.

Date	Opening price	Closing price	Amount
2019/12/4	2876.9079	2878.1157	136000000000
2019/12/3	2869.8822	2884.6988	135000000000
2019/12/2	2874.4484	2875.8072	139000000000
2019/11/29	2885.9744	2871.9813	140000000000
2019/11/28	2902.3644	2889.6934	123000000000
...



3.2 Get the sequence number of the record with the maximum value and do inter-row calculation



We need to know the sequence number of the record corresponding to the highest closing price, and then compare it with the previous trading day to get the result.

SPL is as follows, where pmax function is used to get the sequence number of record with the maximum value:

	A	B
1	=file("000001.csv").import@ct()	/Import the file
2	=A1.select(YEAR(Date)==2019)	/Select records of 2019
3	=A2.pmax('Closing price')	/Get the sequence number of the record with the highest closing price
4	=A2.calc(A3, 'Closing price'/'Closing price'[-1]-1)	/Divide the closing price of the current day by that of the previous day to calculate the growth rate

Similarly, you can use the pmin function to get the sequence number of the record corresponding to the minimum value:

	A	B
3	=A2.pmin('Closing price')	/Get the sequence number of the record with the lowest closing price

✦ 3.2 Get the sequence number of the record with the maximum value and do inter-row calculation



The maximum value is not necessarily unique. If you want to return sequence numbers of all eligible records, just use @a option with the pmax function:

	A	B
3	=A2.pmax@a('Closing price')	/Get the sequence numbers of all the records with the highest closing price

If you want to get the records from back to front, just use the @z option with pmax function:

	A	B
3	=A2.pmax@z('Closing price')	/Get the sequence number of the record with the highest closing price from back to front

✦ 3.3 Get another field value of the record with the maximum value



Get the record corresponding to the maximum value of a specified field in the table and then obtain another field value of the record.

Based on the score table, find the student ID with the highest math score in class one.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 3.3 Get another field value of the record with the maximum value



SPL is as follows, where maxp function is used to get the record with the maximum value, and then obtain the desired student ID:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Scores where SUBJECT='Math' and CLASS='Class one'")	/Query the math scores of class one
3	=A2.maxp(SCORE)	/Get the record with the highest score
4	=A3.STUDENTID	/Get the student ID

Similarly, the minp function is used to get the record with the minimum value:

	A	B
3	=A2.minp(SCORE)	/Get the record with the lowest score

Both maxp function and minp function support @a and @z options.

✦ 3.4 Find top N field values



Find the first/last N values of the specified field in a table.

Based on the score table below, find the highest three math scores in class one.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 3.4 Find top N field values



The SPL is as follows, where top function is used to get done the task. The negative parameter N means getting scores in descending order.

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Scores where SUBJECT='Math' and CLASS='Class one'")	/Query the math scores of class one
3	=A2.top(-3, SCORE)	/Get the highest 3 scores

The return value is a sequence of scores:

Members
97
97
90

✦ 3.4 Find top N field values



To get the four lowest math scores, just set parameter N as the positive 4 get values in ascending order:

	A	B
3	=A2.top(4, SCORE)	/Get the four lowest scores

The return value is a sequence of scores:

Members
60
60
63
63

✦ 3.5 Get the sequence numbers of records with top N values of a specified field



Get the sequence numbers of records with the first/last N values of a certain field in the table, and do inter-row calculations.

According to the stock trading table, calculate the growth rate of trading amount compared with the previous day for each of the three days with the highest closing prices in 2019.

Date	Opening price	Closing price	Amount
2019/12/4	2876.9079	2878.1157	136000000000
2019/12/3	2869.8822	2884.6988	135000000000
2019/12/2	2874.4484	2875.8072	139000000000
2019/11/29	2885.9744	2871.9813	140000000000
2019/11/28	2902.3644	2889.6934	123000000000
...

✦ 3.5 Get the sequence numbers of records with top N values of a specified field



We need to know the sequence numbers of the records that corresponding to the highest three closing prices, and then compare each with the previous trading day to get the result.

SPL is as follows, where the ptop function is used to get the sequence numbers of the records with the highest three closing prices:

	A	B
1	=file("000001.csv").import@ct()	/Import data file
2	=A1.select(year(Date)==2019)	/Select records of 2019
3	=A2.ptop(-3, 'Closing price')	/Get the sequence numbers of the records with top3 closing prices
4	=A3.(A2(~).Amount/A2(~+1).Amount-1)	/Recursively divide the trading amount of the current day by that of the previous day to get the growth rate

A3's result:

VALUE
154
156
157

A4's result:

VALUE
-0.02777777777777779
-0.013888888888888884
0.011235955056179803

✦ 3.5 Get the sequence numbers of records with top N values of a specified field



To get the sequence numbers of records with two lowest closing prices.

	A	B
1	=file("000001.csv").import@ct()	/Import data file
2	=A1.select(YEAR(Date)==2019)	/Select records of 2019
3	=A2.ptop(2, 'Closing price')	/Get the sequence numbers of records with two lowest closing prices

By default, the ptop function returns a sequence of sequence numbers. Below is A3's result:

VALUE
224
225

✦ 3.5 Get the sequence numbers of records with top N values of a specified field



The ptop function can be used to find the sequence number of the record with the highest closing price. The SPL script is as follows:

	A	B
1	=file("000001.csv").import@ct()	/Import data file
2	=A1.select(YEAR(Date)==2019)	/Select records of 2019
3	=A2.ptop@1(-1, 'Closing price')	/Get the sequence number of the record with the highest closing price

When N is ± 1 , the ptop@1(n, x) function returns a numeric value, which is the same with the pmax & pmin functions.

Value
154

✦ 3.6 Get records with top N values in a specified field



Get the records corresponding to the first / last N values of a certain field in the table.

Based on the score table below, find records of the students whose math scores rank top three in class one.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 3.6 Get records with top N values in a specified field



In SPL, top(n; x) function can get the records with the top N values. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Scores where SUBJECT='Math' and CLASS='Class one'")	/Query the math scores of class one
3	=A2.top(-3; SCORE)	/Get the records of top 3

A3's result:

CLASS	STUDENTID	SUBJECT	SCORE
Class one	13	Math	97
Class one	10	Math	97
Class one	7	Math	90

✦ 3.7 Get other field values of the records with top N values of a specified field



Get other field values of the records corresponding to top N values of a specified field in a table.

Based on the login log table, find a certain user's IP address at his/her first login.

ID	USERID	IP	LOGINTIME
1	1	37.17.184.11	2012/05/09 09:01:10
2	3	61.134.201.1	2012/05/09 09:02:43
3	7	124.114.171.101	2012/05/09 09:03:18
4	2	183.202.48.25	2012/05/09 09:05:15
5	15	1.24.216.5	2012/05/09 09:05:55
...

✦ 3.7 Get other field values of the records with top N values of a specified field



We can use Oracle's keep function to get this done:

```
select
    min(IP) keep (dense_rank first order by LOGINTIME) IP
from
    LoginLog
where
    USERID=8
```

Execution result:

IP
223.223.118.1

✦ 3.7 Get other field values of the records with top N values of a specified field



In fact, we only need the record that the user logged in for the first time. In SPL, top(n; x) function can get it in a clear way. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from LoginLog where USERID=8")	/Query login records where the user ID is 8
3	=A2.top(1; LOGINTIME)	/Get the record of the first login
4	=A3.IP	/Get the IP field value

A3's result:

ID	USERID	IP	LOGINTIME
8	8	223.223.118.1	2012/05/09 09:07:17

A4's result:

IP
223.223.118.1

✦ 3.8 Get top N records in each group after grouping



Group records in a table and get the records that corresponding to Top N values of the specified field from each group.

Based on the score table below, for each subject, find the IDs of students whose scores rank top 2 in each class.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 3.8 Get top N records in each group after grouping



SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Scores")	/Query <i>Scores</i> table
3	=A2.group(CLASS,SUBJECT;~.top(-2;SCORE):TOP2)	/Group by class and subject and get records with the top two scores in each group
4	=A3.(TOP2).conj()	/Concatenate all the top two records obtained in A3

A4's result:

CLASS	STUDENTID	SUBJECT	SCORE
Class one	4	English	96
Class one	9	English	93
Class one	13	Math	97
Class one	10	Math	97
...

✦ 3.8 Get top N records in each group after grouping




A3:group function group records by class and subject . Here getting top N is an aggregate operation, which finds the first two from each subset.

	A	B
3	A3: =A2.group(CLASS,SUBJECT;~.top(-2;SCORE):TOP2)	/Group by class and subject, then get records with top two scores in each group

A3's result:

CLASS	SUBJECT	Members
Class one	English	[[Class one,4,English,96],[Class one,9,English,93]]
Class one	Math	[[Class one,13,Math,97],[Class one,10,Math,97]]
...



CLASS	STUDENTID	SUBJECT	SCORE
Class one	13	Math	97
Class one	10	Math	97

✦ 3.9 Perform grouping & aggregation and get top N records in each group



Perform grouping & aggregation and then get records that corresponding to the first / last N values of the specified field from each group.

Based on *the EMPLOYEE table*, find employees whose salaries rank top three in each department.

EID	NAME	DEPT	SALARY
1	Rebecca	R&D	7000
2	Ashley	Finance	11000
3	Rachel	Sales	9000
4	Emily	HR	7000
5	Ryan	R&D	13000
...

✦ 3.9 Perform grouping & aggregation and get top N records in each group



SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>EMPLOYEE</i> table
3	=A2.groups(DEPT; top(-3;SALARY):TopSalary)	/Group by department and get records with salaries ranking top 3 in each group
4	=A3.(TopSalary).conj()	/Concatenate eligible records of every department

A4's result:

EID	NAME	DEPT	SALARY
20	Alexis	Administration	16000
42	Michael	Administration	12000
18	Jonathan	Administration	7000
2	Ashley	Finance	11000
32	Andrew	Finance	11000
...

A.groups() function calculates in a cumulative way and does not generate the intermediate grouping subsets.



Chapter 4

SPL
COOKBOOK

Grouping & Aggregation

✦ 4.1 Aggregation operation: SUM



Aggregate data in a table to get the sum.

Below is a table recording cities' GDP. Now calculate GDP per capita of municipalities, the first-tier cities and the second-tier cities respectively.

ID	City	GDP	Population
1	Shanghai	32679	2418
2	Beijing	30320	2171
3	Shenzhen	24691	1253
4	Guangzhou	23000	1450
5	Chongqing	20363	3372
...

✦ 4.1 Aggregation operation: SUM



SPL script is as follows, where sum() function is used to calculate sum.

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from GDP")	/Query <i>GDP</i> table
3	[["Beijing", "Shanghai", "Tianjing", "Chongqing"].pos(?)>0, ["Beijing", "Shanghai", "Guangzhou", "Shenzhen"].pos(?)>0, ["Chengdu", "Hangzhou", "Chongqing", "Wuhan", "Xian", "Suzhou", "Tianjing", "Nanjing", "Changsha", "Zhengzhou", "Dongguan", "Qingdao", "Shenyang", "Ningbo", "Kunming"].pos(?)>0]	/Enumerate municipalities, the first-tier cities and the second-tier cities
4	=A2.enum@r(A3,City)	/Enumeration grouping by city
5	=A4.new(A3(#):Area,~.sum(GDP)/~.sum(Population)*10000:CapitaGDP)	/Calculate GDP per capita of each group

A5	Area	CapitaGDP
	["Beijing", "Shanghai", "Tianjing", "Chongqing"].pos(?)>0	107345.03
	["Beijing", "Shanghai", "Guangzhou", "Shenzhen"].pos(?)>0	151796.49
	["Chengdu", "Hangzhou", "Chongqing", "Wuhan", "Xian", "Suzhou", "Tianjing", "Nanjing", "Changsha", "Zhengzhou", "Dongguan", "Qingdao", "Shenyang", "Ningbo", "Kunming"].pos(?)>0	106040.57

✦ 4.2 Aggregation operation: MAX & MIN



Aggregate data in a table to calculate the maximum value or minimum value.

Merge the orders records of customer ANATR that have overlapping time periods.

OrderID	Customer	SellerId	OrderDate	FinishDate
10308	ANATR	7	2012/09/18	2012/10/16
10309	ANATR	3	2012/09/19	2012/10/17
10625	ANATR	3	2013/08/08	2013/09/05
10702	ANATR	1	2013/10/13	2013/11/24
10759	ANATR	3	2013/11/28	2013/12/26
...

✦ 4.2 Aggregation operation: MAX & MIN



SPL script is as follows, where max() function is used to calculate the maximum value and min() function to calculate the minimum value:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Orders where Customer='ANATR' order by OrderDate")	/Select orders records of customer ANATR and sort them by order date
3	=A2.group@i(OrderDate>max(FinishDate[,-1]))	/If the current order date is later than all previous finish dates, put the record to a new group
4	=A3.new(Customer,~.min(OrderDate):OrderDate,~.max(FinishDate):FinishDate)	/Make the earliest order date in each group the new order date and the latest finish date the new finish date

A3
Member
[[10308,ANATR,7,...], [10309,ANATR,3,...]]
[[10625,ANATR,3,...]]
[[10702,ANATR,1,...]]
[[10759,ANATR,3,...], [11079,ANATR,7,...]]
...

A4	Customer	OrderDate	FinishDate
	ANATR	2012/09/18	2012/10/17
	ANATR	2013/08/08	2013/09/05
	ANATR	2013/10/13	2013/11/24
	ANATR	2013/11/28	2013/12/29

✦ 4.3 Aggregation operation: AVERAGE



We need to calculate the average value during a transposition. Below is part of the data in the EMPLOYEE table:

ID	NAME	DEPT	STATE	SALARY
1	Rebecca	R&D	California	7000
2	Ashley	Finance	New York	11000
3	Rachel	Sales	New Mexico	9000
4	Emily	HR	Texas	7000
...

Calculate the average salary of each department in different states and display the result set in the following format:

DEPT	California	Florida	New York	Texas	...
Finance	8000	10000	7500	8166.67	...
HR	10000	7000	5000	6500	...
...

✦ 4.3 Aggregation operation:AVERAGE



SPL script is as follows, where avg() function is used to calculate the average value:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>EMPLOYEE</i> table
3	=A2.groups(DEPT,STATE;avg(SALARY):AvgSalary)	/Grouping & aggregation; avg function calculates the average salary of each department in each region
4	=A3.pivot(DEPT;STATE, AvgSalary)	/Transpose data according to the target table

A4					
DEPT	California	Florida	New York	Texas	...
Finance	8000	10000	7500	8166.67	...
HR	10000	7000	5000	6500	...
...

✦ 4.4 Aggregation operation:COUNT



Group records and count numbers base on the values of a target field.
For each subject, find the number of students in class one who failed.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 4.4 Aggregation operation:COUNT



SPL script is as follows, where count() function is used to count the total number:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Scores where CLASS='Class one'")	/Query scores of students in class one
3	=A2.groups(SUBJECT; count(SCORE<60):FailCount)	/Grouping & aggregation, in which the count function counts the number of students whose scores are less than 60 for each subject

A3	SUBJECT	FailCount
	English	2
	Math	0
	PE	2

✦ 4.5 Aggregation operation: logical AND



Perform logical AND over a sequence of Boolean values.

According to a set of Excel files that records the types of terminals the students in a primary school used during online learning, find whether all the students use mobile phones. Below are the questionnaire of each class and the table:

▼	Primary School
>	Grade1
>	Grade2
▼	Grade3
	Class1
	Class2
	Class3
	Class4
	Class5
	Class6
>	Grade4
>	Grade5
>	Grade6

ID	STUDENT_NAME	TERMINAL
1	Rebecca Moore	Phone
2	Ashley Wilson	Phone,PC,Pad
3	Rachel Johnson	Phone,PC,Pad
4	Emily Smith	Phone,Pad
5	Ashley Smith	Phone,PC
6	Matthew Johnson	Phone
7	Alexis Smith	Phone,PC
8	Megan Wilson	Phone,PC,Pad
...

✦ 4.5 Aggregation operation: logic AND



A.ifn() function gets the first non-null member; A.cand() function performs the logical AND operation on members. SPL script is as follows:

	A	B	C
1	=directory@ps("D:/Primary School")		/Recursively traverse directories to list all files
2	for A1	=file(A2).xlsimport@t()	/Recursively Import the questionnaire Excel files of all classes
3		=B2.([TERMINAL,"Phone"].ifn()).split@c().pos("Phone") > 0)	/A null terminal value does not mean that the mobile is not used; and ifn() function is used to ensure the judgement is true
4	=B3.cand()		/cand() function judges whether all members of B3 are true

A1
Member
D:\Primary School\Grade1\Class1\Questionnaire.xlsx
D:\Primary School\Grade1\Class2\Questionnaire.xlsx
D:\Primary School\Grade1\Class3\Questionnaire.xlsx
...

B3
Member
true
true
true
...

A4
Value
false

✦ 4.6 Aggregation operation:logic OR



Perform logical OR over a sequence of Boolean values.

Query whether customer RATTC ranked top three in terms of monthly sales amounts in 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 4.6 Aggregation operation:logic OR



SPL script is as follows, where A.cor() function is used to perform logical OR operation on members:

	A	B
1	=connect("db").query("select * from sales")	/Connect to database and query sales table
2	=A1.select(year(OrderDate)=2014)	/Select sales records of 2014
3	=A2.group(month(OrderDate))	/Use group function to group records of 2014 by month
4	=A3.(~.groups(Customer; sum(Amount):Amount))	/Group records in each month by customer and calculate sales amount for each group
5	=A4.new(~.top(-3; Amount):Top3)	/Traverse records of every month to get the top 3 customers in terms of sales amount
6	=A5.(Top3.(Customer).pos("RATTC")>0)	/Judge whether the top 3 of each month include customer RATTC
7	=A6.cor()	/Use cor() function to find whether a "true" exists in A6

A6	Member	A7	Value
	false		true
	false		
	true		
	...		

✦ 4.7 Aggregation operation: Count distinct values



Perform count operation over distinct values in a table sequence.

Find the most suitable field in the following data file for the primary key.

PassengerId	Survived	Pclass	Name	Sex	Age
1	0	3	"Braund, Mr. Owen Harris"	male	22
2	1	1	"Cumings, Mrs. John Bradley"	female	38
3	1	3	"Heikkinen, Miss. Laina"	female	26
4	1	1	"Futrelle, Mrs. Jacques Heath"	female	35
5	0	3	"Allen, Mr. William Henry"	male	35
6	0	3	"Moran, Mr. James"	male	
7	0	1	"McCarthy, Mr. Timothy J"	male	54
...

✦ 4.7 Aggregation operation: Count distinct values



SPL script is as follows, in which the `icount()` function is to count distinct values:

	A	B	C	D
1	<code>=file("titanic_train.csv").import@cqt()</code>			/Read the data file
2	<code>=A1.fno().new(A1.fname(~):Name,A1.field(~).icount():DCount)</code>			/Use the <code>icount()</code> function to count the number of distinct values in each field
3	<code>=A2.select(DCount==A1.len())</code>			/Select the field where the number of distinct members is equivalent with the count of all members
4	<code>if (A3.len() > 1)</code>	<code>=A3.select(like@c(Name,"*id*"))</code>		/If there are more than one eligible field, select those whose names contain string id
5		<code>if (B4.len() > 0)</code>	<code>>A3=B4</code>	/If there is such a field, assign it to A3
6	<code>=A3.minp(len(Name)).Name</code>			/Select the field that has the shortest name

A6	Value
	PassengerId

✦ 4.8 Aggregation operation:MEDIAN



Group members and calculate the median for each group.

Based on the employee salary table, calculate the median salary of each department.

ID	NAME	DEPT	SALARY
1	Rebecca	R&D	7000
2	Ashley	Finance	11000
3	Rachel	Sales	9000
4	Emily	HR	7000
5	Ashley	R&D	16000
...

✦ 4.8 Aggregation operation:MEDIAN



SPL script is as follows, where the median() function is used to get median:

	A	B
1	=connect("db").query("select * from employee")	/Connect to database and query <i>employee</i> table
2	=A1.groups(DEPT;median(,SALARY):MedianSalary)	/Group and aggregate by department to calculate the median salary of each department

A3	DEPT	MedianSalary
	Administration	9500.0
	Finance	7000.0
	HR	7000
	Marketing	7000
	Production	6500

✦ 4.9 Aggregation operation: RANKING



Group members and get the ranking for each group.

Find the ranking of the total score of the student in class one whose student ID is 8.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 4.9 Aggregation operation:RANKING



SPL script is as follows, where A.rank() function is used to calculate the ranking:

	A	B
1	=connect("db").query("select * from SCORES where CLASS='Class one'")	/Connect the data source and query scores of class one
2	=A1.groups(STUDENTID;sum(SCORE):TotalScore)	/Group and sum the total score of each student
3	=A2.select(STUDENTID==8).TotalScore	/Get the total score of the student whose id is 8
4	=A2.rank@z(A3, TotalScore)	/Use A.rank() function to calculate the ranking, where @z option ranks total scores in descending order

A4	Value
	13

✦ 4.10 Aggregation operation: An application scenario of RANKING



According to the test results of data scoring, calculate the AUC index of the data mining model through the aggregate ranking operation.

ID	Predicted_Y	Original_Y
10	0.012388464367608093	0.0
11	0.01519899123978988	0.0
13	0.0007920238885061248	0.0
19	0.0012656367468159102	0.0
21	0.009460545997473379	0.0
...

✦ 4.10 Aggregation operation: An application scenario of RANKING



SPL script is as follows, where ranks() function is used to get the ranking:

	A	B
1	=file("p.csv").import@ct()	/Import the file
2	=P=A1.pselect@a(Original_Y==1),M=P.len()	/Select all records whose original target is 1, set their row number as P and quantity as M
3	=N=A1.len()-M	/For records whose original target is not 1, set their quantity as N
4	=A1.(Predicted_Y).ranks@s()	/Use ranks() function to calculate the ranking of scoring values. @s option is used to get the average of rankings that have equal values
5	=(A4(P).sum()-M*(1+M)/2)/(M*N)	/Calculate AUC value according to the formula

A5	Value
	0.9055583178459794



Chapter 5

SPL COOKBOOK

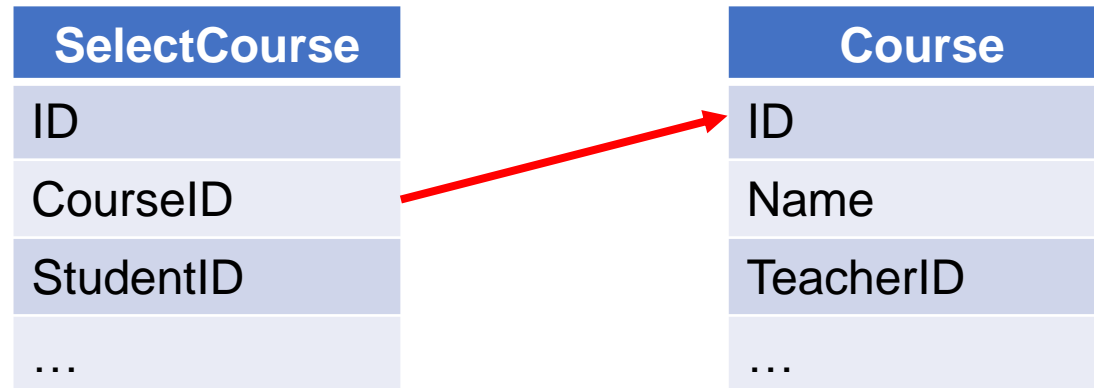
Alignment grouping

✦ 5.1 Group by the specified order, each group keeps only one record



Find unreferenced records based on two associated tables.

According to the associated *Course* table and *SelectCourse* table, query which courses are not selected according to the order of *Course* table.



✦ 5.1 Group by the specified order, each group keeps only one record



SPL script is as follows, where align(A:x, y) function is used to perform alignment grouping:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from SelectCourse")	/Query <i>SelectCourse</i> table
3	=A1.query("select * from Course")	/Query <i>Course</i> table
4	=A2.align(A3:ID,CourseID)	/Align <i>SelectCourse</i> table by Course.ID and select one matching member for each group
5	=A3(A4.pos@a(null))	/Select records of unselected courses (value is null) in the <i>Course</i> table

A5's result:

ID	NAME	TeacherID
1	Environmental protection and sustainable development	5
10	Music appreciation	18

A4's result:

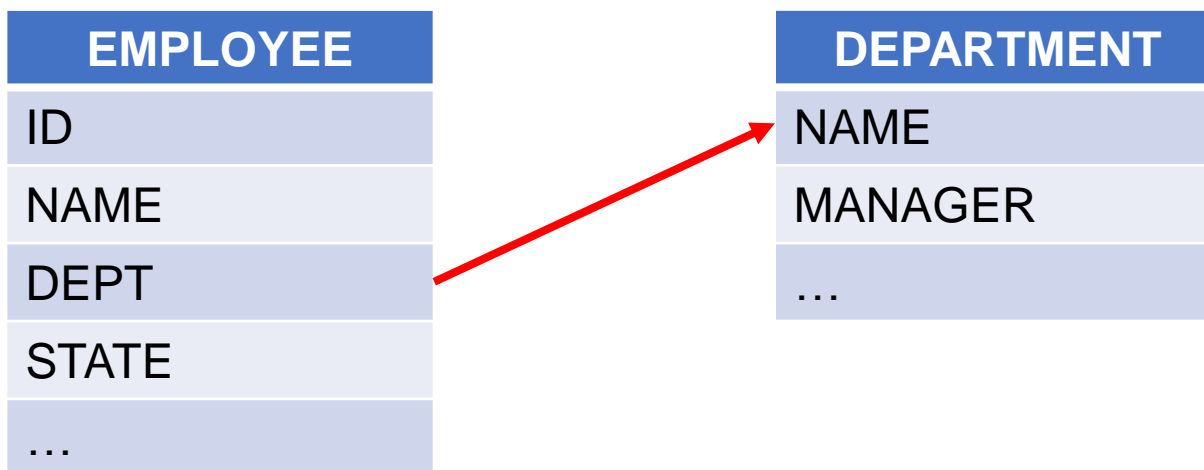
Members
(null)
[13,2,7]
[7,3,41]
[45,4,28]
[3,5,52]
[1,6,59]
[10,7,13]
[8,8,49]
[6,9,57]
(null)

✦ 5.2 Group in specified order



Group and perform sum over all records in a table according to the order of a specified field in another table.

Based on the associated *EMPLOYEE* table and *DEPARTMENT* table, calculate the number of employees in each department by the department order in *DEPARTMENT* table.



✦ 5.2 Group in specified order



SPL script is as follows, where align@a option is used:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>EMPLOYEE</i> table
3	=A1.query("select * from DEPARTMENT")	/Query <i>DEPARTMENT</i> table
4	=A2.align@a(A3:ID, DEPARTMENT)	/Align and group <i>EMPLOYEE</i> table by DEPARTMENT.ID; @a option returns all matching records for each group
5	=A4.new(DEPT, ~.count():COUNT)	/Count the number of employees in each department

✦ 5.2 Group in specified order



A4's result:

Members	ID	NAME	DEPT	STATE
[[18,Jonathan,Admin,...], [20,Alexis, Admin,...], ...]	1	Rebecca	R&D	California
[[1,Rebecca,R&D,...],[5,Ashley,R&D,...],...]	5	Ashley	R&D	Texas
[[3,Rachel,Sales,...],[6,Matthew,Sales,...],...]	10	Ryan	R&D	Pennsylvania
...

A5's result:

DEPT	COUNT
Admin	4
R&D	29
Sales	187
...	...

✦ 5.3 Group in specified order and put unmatched records in a new group



Group records in a table according to the order of specified values in a field; put unmatched records in a new group.

Based on the employee salary table, calculate the average salary of states according to the order specified by the sequence [California, Texas, New York, Florida], and put the average salary of all the other states in "Other".

ID	NAME	STATE	SALARY
1	Rebecca	California	7000
2	Ashley	New York	11000
3	Rachel	New Mexico	9000
4	Emily	Texas	7000
5	Ashley	Texas	16000
...

✦ 5.3 Group in specified order and put unmatched records in a new group



SPL script is as follows, where @an options are used with align() function:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>EMPLOYEE</i> table
3	[California,Texas,New York,Florida]	/Create a sequence of states
4	=A2.align@an(A3,STATE)	/Align and group, <i>EMPLOYEE</i> table is grouped by the specified states; @a option returns all matching records, and @n option puts mismatched records in a new group
5	=A4.new(if (#>A3.len(),"Other", STATE):STATE, ~.avg(SALARY):AvgSalary)	/Calculate the average salary of each group and generate a new table sequence. Rename the last group "Other"; by default it is the state name of the first record of this group

✦ 5.3 Group in specified order and put unmatched records in a new group



A4's result:

Members	ID	NAME	STATE	SALARY
[[1,Rebecca,California,...], [6,Matthew,California,...], ...]	3	Rachel	New Mexico	9000
[[4,Emily,Texas,...],[5,Ashley,Texas,...],...]	7	Alexis	Illinois	9000
[[2,Ashley,New York,...],[12,Jessica,New York,...],...]	10	Ryan	Pennsylvania	13000
[[13,Daniel, Florida,...],[14,Alyssa,Florida,...],...]	19	Samantha	Pennsylvania	10000
[[3,Rachel,New Mexico,...],[7,Alexis,Illinois,...],...]

A5's result:

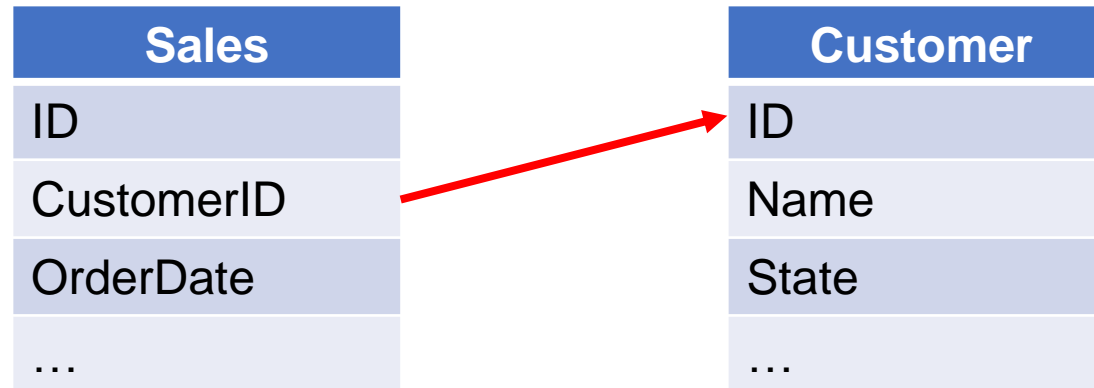
STATE	SALARY
California	7700.0
Texas	7592.59
New York	7677.77
Florida	7145.16
Other	7308.1

✦ 5.4 Group by sequence number, each group keeps only one record



Find unreferenced records based on two associated tables.

According to the associated *Sales* table and *Customer* table, list customers without sales record in 2014 in the order of their IDs.



✦ 5.4 Group by sequence number, each group keeps only one record



SPL script is as follows, where `align(n, y)` function is used to perform alignment grouping:

	A	B
1	<code>=connect("db")</code>	<code>/Connect to database</code>
2	<code>=A1.query("select * from Sales")</code>	<code>/Query Sales table</code>
3	<code>=A1.query("select * from Customer")</code>	<code>/Query Customer table</code>
4	<code>=A3.(ID)</code>	<code>/Select customer IDs from Customer table</code>
5	<code>=A2.align(A4.len(), A4.pos(CustomerID))</code>	<code>/Align and group Sales table by sequence numbers of customers</code>
6	<code>=A3(A5.pos@a(null))</code>	<code>/Select records of customers without sales record (value is null) from Customer table</code>

A6 result:

ID	Name	State	...
ALFKI	CMA-CGM	Texas	...
CENTC	Nedlloyd	Florida	...

✦ 5.5 Group by number



Group all records in a table by number and sum them up.

According to the *Orders* table, list in order the total number of orders per month in 2013.

ID	CustomerID	OrderDate	Amount
10248	VINET	2012/07/04	428.0
10249	TOMSP	2012/07/05	1842.0
10250	HANAR	2012/07/08	1523.5
10251	VICTE	2012/07/08	624.95
10252	SUPRD	2012/07/09	3559.5
...

✦ 5.5 Group by number



SPL script is as follows, where @a option is used with align(n, y) function to select all matching members in each group:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Orders where year(OrderDate)=2013")	/Query orders in 2013
3	=A2.align@a(12,month(OrderDate))	/Align and divide orders in 2013 into 12 groups by the order month; @a option selects all matching members for each group
4	=A3.new(#:Month,~.count():OrderCount)	/List the total number of orders per month

✦ 5.5 Group by number



A3's result:

Members
[[10400,EASTC,1],[10401,RATTC,1],...]
[[10433,PRINI,2],[10434,FOLCO,2],...]
[[10462,CONSH,3],[10463,SUPRD,3],...]
[[10492,BOTTM,4],[10493,LAMAI,4],...]
[[10523,SEVES,5],[10524,BERGS,5],...]
[[10555,SAVEA,6],[10556,SIMOB,6],...]
[[10585,WELLI,7],[10586,REGGC,7],...]
[[10618,MEREP,8],[10619,MEREP,8],...]
[[10651,WANDK,9],[10652,WANDK,9],...]
[[10688,VAFFE,10],[10689,BERGS,10],...]
[[10726,EASTC,11],[10727,REGGC,11],...]
[[10760,MAISD,12],[10761,RATTC,12],...]

A4's result:

Month	OrderCount
1	33
2	29
3	30
4	31
5	32
6	30
7	33
8	33
9	37
10	38
11	34
12	48

✦ 5.6 Repeatedly grouped by sequence numbers



Align and group records by a calculated sequence number array, during which one record could match more than one group.

Group records by label and count the frequencies of each label based on *PostRecord* table.

ID	TITLE	Author	Label
1	Easy analysis of Excel	2	Excel,ETL,Import,Export
2	Early commute: Easy to pivot excel	3	Excel,Pivot,Python
3	Initial experience of SPL	1	Basics,Introduction
4	Talking about set and reference	4	Set,Reference,Dispersed,SQL
5	Early commute: Better weapon than Python	4	Python,Contrast,Install
...

✦ 5.6 Repeatedly grouped by sequence numbers



SPL script is as follows, where @r option is used with align(n, y) function; each member may match multiple groups:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from PostRecord")	/Query <i>PostRecord</i> table
3	=A2.conj(Label.split(",")).id()	/Separate labels by commas and merge them into a sequence to obtain all distinct labels
4	=A2.align@ar(A3.len(),A3.pos(Label.split(",")))	/Use @r option with align function to align and group posts by label positions repeatedly
5	=A4.new(A3(#):Label,~.count():Count).sort@z(Count)	/Count the posts under each label and arrange them in descending order

A5's result:

Label	Count
SPL	7
SQL	6
Basics	5
...	...

✦ 5.7 Group by segments of field values



Group records by segments of values of the specified field and count records in each group. Based on the salary table below, count the employees in three salary ranges respectively (<8000 , ≥ 8000 and <12000 , > 12000).

ID	NAME	BIRTHDAY	SALARY
1	Rebecca	1974-11-20	7000
2	Ashley	1980-07-19	11000
3	Rachel	1970-12-17	9000
4	Emily	1985-03-07	7000
5	Ashley	1975-05-13	16000
...

✦ 5.7 Group by segments of field values



pseg(x) function is used in align(n,y) function to locate a segment of records. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>EMPLOYEE</i> table
3	[0,8000,12000]	/Define salary ranges
4	=A2.align@a(A3.len(),A3.pseg(SALARY))	/Use pseg() function to get the corresponding salary range
5	=A4.new(A3 (#):SALARY,~.count():COUNT)	/Count the employees in each group

A5's result:

SALARY	COUNT
0	308
8000	153
12000	39

✦ 5.8 Group by segment according to expression result



Group records by segment according to the calculation result of expression and calculate average of each group.

Calculate average salary for employees who have been with the company less than 10 years, 10 to 20 years and not less than 20 years respectively, based on the *EMPLOYEE* table.

ID	NAME	HIREDATE	SALARY
1	Rebecca	2005-03-11	7000
2	Ashley	2008-03-16	11000
3	Rachel	2010-12-01	9000
4	Emily	2006-08-15	7000
5	Ashley	2004-07-30	16000
...

✦ 5.8 Group by segment according to expression result



In the align(n,y) function, pseg(x) function is used to locate a segment of records. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>EMPLOYEE</i> table
3	[0,10,20]	/Define intervals of stay
4	=now()	/Get the current date and time
5	=A2.align@a(A3.len(),A3.pseg(elapse@y(A4,-~), HIREDATE))	/Use pseg() function to get the corresponding interval for each hire date
6	=A5.new(A3(#):EntryYears,~.avg(SALARY):AvgSalary)	/Calculate the average salary of each group

A6 result:

EntryYears	AvgSalary
0	6777.78
10	7445.53
20	6928.57

✦ 5.9 Group by enumerated conditions, records are not repeatedly grouped



Group records according to the enumerated conditional expression. One record will only be placed in the first matching group.

Classify cities by population based on *UrbanPopulation* table.

ID	City	Population	Province
1	Shanghai	12286274	Shanghai
2	Beijing	9931140	Beijing
3	Chongqing	7421420	Chongqing
4	Guangzhou	7240465	Guangdong
5	Hong Kong	7010000	Hong Kong Special Administrative Region
...

✦ 5.9 Group by enumerated conditions, records are not repeatedly grouped



SPL script is as follows, where enum function is used to implement enumeration grouping:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from UrbanPopulation")	/Query <i>UrbanPopulation</i> table
3	[?>2000000,?>1000000,?>500000,?<=500000]	/Cities are classified by population range
4	=A2.enum(A3,Population)	/enumeration grouping according to the condition defined by A3

A4's result:

Members	ID	City	Population	Province
[[1,Shanghai,12286274,...], [2,Beijing, 9931140,...], ...]	1	Shanghai	12286274	Shanghai
[[28,Changsha,1965282,...], [29,Nanchang,1900817,...], ...]	2	Beijing	9931140	Beijing
[[69,Huainan,974026,...], [70,Haikou, 967336,...], ...]	3	Chongqing	7421420	Chongqing
[]

✦ 5.10 Group by enumerated conditions, unmatched records are put in a new group



Group records according to the enumerated conditional expression, and put the unmatched records into a new group.

Group employees by age conditions [less than 35 years old, less than 45 years old], and calculate the average salary of each group based on *EMPLOYEE* table. Those who do not meet the conditions are put into a new group.

ID	NAME	BIRTHDAY	SALARY
1	Rebecca	1974-11-20	7000
2	Ashley	1980-07-19	11000
3	Rachel	1970-12-17	9000
4	Emily	1985-03-07	7000
5	Ashley	1975-05-13	16000
...

✦ 5.10 Group by enumerated conditions, unmatched records are put in a new group



SPL script is as follows, where @n option is used with enum() function to put unmatched members to a new group:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from EMPLOYEE")	/Query <i>EMPLOYEE</i> table
3	[?<35,?<45]	/Divide employee ages into <35, <45
4	=A2.enum@n(A3, age(BIRTHDAY))	/First, calculate age according to the birthday; then group records by the enumerated age intervals, during which the unmatched ones are put in a new group
5	=A4.new(if (#>A3.len(), "Other",A3(#)):AGE,~.avg(SALARY):AvgSalary)	/Calculate the average salary of each group, and set the name of the last group as Other

A5's result:

AGE	AvgSalary
?<35	7118.18
?<45	7448.16
Other	7395.06

✦ 5.11 Repeatedly group by enumerated conditions



Group and calculate records according to multiple specified sequences; records can be repeatedly grouped.

Calculate the GDP per capita of municipalities, the first-tier cities and the second-tier cities respectively based on *GDP* table; a member may be put into more than one group.

ID	City	GDP	Population
1	Shanghai	32679	2418
2	Beijing	30320	2171
3	Shenzhen	24691	1253
4	Guangzhou	23000	1450
5	Chongqing	20363	3372
...

✦ 5.11 Repeatedly group by enumerated conditions



SPL script is as follows, where enum@r option is used to check whether all members match in each group:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from GDP")	/Query GDP table
3	[["Beijing", "Shanghai", "Tianjing", "Chongqing"].pos(?)>0, ["Beijing", "Shanghai", "Guangzhou", "Shenzhen"].pos(?)>0, ["Chengdu", "Hangzhou", "Chongqing", "Wuhan", "Xian", "Suzhou", "Tianjing", "Nanjing", "Changsha", "Zhengzhou", "Dongguan", "Qingdao", "Shenyang", "Ningbo", "Kunming"].pos(?)>0]	/Enumerate municipalities, the first-tier cities and the second-tier cities
4	=A2.enum@r(A3, City)	/Group by enumerated group of cities
5	=A4.new(A3(#):Area, ~.sum(GDP)/~.sum(Population)*10000:CapitaGDP)	/Calculate GDP per capita of each group

A5's result:

Area	CapitaGDP
["Beijing", "Shanghai", "Tianjing", "Chongqing"].pos(?)>0	107345.03
["Beijing", "Shanghai", "Guangzhou", "Shenzhen"].pos(?)>0	151796.49
["Chengdu", "Hangzhou", "Chongqing", "Wuhan", "Xian", "Suzhou", "Tianjing", "Nanjing", "Changsha", "Zhengzhou", "Dongguan", "Qingdao", "Shenyang", "Ningbo", "Kunming"].pos(?)>0	106040.57



Chapter 6

SPL COOKBOOK

Subsets after grouping

✦ 6.1 Inter-row calculation in subsets after grouping



Group records according to a field in a table, and perform inter-row calculation in each group.
Based on the user consumption table, for each user calculate the difference between the last expense and the previous expense.

User consumption table

Index	SEQ	USERID	PAYTIME	PAYAMOUNT
1	1	19660178411	2013-07-04 01:...	618.939
2	2	19118341234	2011-03-15 16:...	528.155
3	3	19181723653	2012-12-21 11:...	231.114
4	4	19199550134	2014-07-15 01:...	685.382
5	5	19860606128	2013-04-27 16:...	922.376
6	6	19459311399	2010-11-25 15:...	311.366
7	7	19890228863	2012-02-26 16:...	2.537
8	8	19251553201	2011-01-26 17:...	723.783
9	9	19470783075	2014-02-06 05:...	662.281

✦ 6.1 Inter-row calculation in subsets after grouping



SPL code

	A
1	=db.query("SELECT * FROM USERPAY")
2	=A1.group@u(USERID)
3	=A2.(~.top(-2;PAYTIME))
4	=A3.new(~.USERID,if(~.count())<2,0,(~(1).PAYAMOUNT-~(2).PAYAMOUNT)):BALANCE)

A2 : group by user, no sorting required

A3: Sort each group by pay time and get the last two records

A4: Calculate the difference

SPL output

Index	USERID	BALANCE
1	19660178411	0
2	19118341234	0
3	19181723653	-85.02600000000001
4	19199550134	-297.96900000000005
5	19860606128	490.84099999999995
6	19459311399	-162.027
7	19890228863	0
8	19251553201	389.68299999999994
9	19470783075	-274.70399999999995

✦ 6.2 Group in the order of record and perform count



When the field value of the next record changes, put it to a new group, and perform count after the grouping finishes.

Based on the following the seating plan table, count the maximum number of continuous empty seats.

Seating plan for a movie theater

Index	🔑 ROW	🔑 COL	EMPTY
1	1	1	<u>NO</u>
2	1	2	<u>NO</u>
3	1	3	<u>YES</u>
4	1	4	<u>YES</u>
5	1	5	<u>NO</u>
6	1	6	<u>NO</u>
7	1	7	<u>NO</u>
8	1	8	<u>NO</u>
9	1	9	<u>NO</u>

✦ 6.2 Group in the order of record and perform count



SPL code

	A
1	=file("D:/CINEMA.ctx").create().cursor().fetch()
2	=A1.group@o(ROW,EMPTY;~.count():CNT)
3	=A2.select(EMPTY=="YES")
4	=A3.max(CNT)

A2: Group records in the current order

A3: Filter away the taken seats

A4: Find the maximum number of continuous empty seats

SPL output

Value
9

✦ 6.3 Ordered conditional grouping



Loop through records in a table to calculate the conditional expression, and create a new group if the result is true.

Based on the seating plan table, count the maximum number of continuous empty seats.

Seating plan for a movie theater

Index	🔑 ROW	🔑 COL	EMPTY
1	1	1	<u>NO</u>
2	1	2	<u>NO</u>
3	1	3	<u>YES</u>
4	1	4	<u>YES</u>
5	1	5	<u>NO</u>
6	1	6	<u>NO</u>
7	1	7	<u>NO</u>
8	1	8	<u>NO</u>
9	1	9	<u>NO</u>

✦ 6.3 Ordered conditional grouping



SPL code

	A
1	=file("D:/CINEMA.ctx").create().cursor().fetch()
2	=A1.group@i(ROW[-1]!=ROW EMPTY[-1]!=EMPTY;EMPTY,~.count():CNT)
3	=A2.select(EMPTY=="YES")
4	=A3.max(CNT)

A2: Group records in the current order by the conditional expression

A3: Filter away the non-empty seats

A4: Find the maximum number of continuous empty seats

SPL output

Value
9

✦ 6.4 Group by number



Group records according to the calculated numbers.

Remove records where the balance is 0 from the account balance table, and group other records by the balance amount every interval of \$500.

Account balance table

Index	ID	AMOUNT
1	19101285231	576.361
2	19102787766	669.582
3	19104055437	0.0
4	19106916106	108.238
5	19107930314	0.0
6	19110297329	342.185
7	19110602563	0.0
8	19110817459	620.286
9	19111537167	0.0

✦ 6.4 Group by sequence number



SPL code

	A
1	=db.query("SELECT * FROM USERACCOUNT WHERE AMOUNT > 0.000001")
2	=A1.group@n(int(AMOUNT/500)+1)

A1: Remove records with a balance of 0

A2: Divide records by balance amounts every interval of \$500

SPL output

Index	Member
1	[[19106916106,108.238],[19110297329,342.185],[...
2	[[19101285231,576.361],[19102787766,669.582],[...
3	[[19112166318,1081.136],[19124900832,1033.40...
4	[[19132788278,1550.4839],[19158122162,1803.9...
5	[[19146667496,2342.6162],[19167335457,2201.6...
6	[[19303492347,2653.645],[19699834396,2529.65...
7	[[19179195162,3258.207],[19256954428,3080.575]]
8	[]
9	[[19566299249,4140.279]]

Grouping by numbers can quickly locate the corresponding subset directly through group numbers. For example, 3000 ~ 3500 corresponds to group 7.



✦ 6.5 Multilevel grouping & aggregation



Each group (a subset) is grouped again.

Based on the daily trading summary table, count the maximum number of days when the closing price of each stock rises consecutively.

Daily trading
summary table

Index	SCODE	TDAY	EPRICE
1	002579	2011-05-06	18
2	002579	2011-05-09	17
3	002579	2011-05-10	17
4	002579	2011-05-11	17
5	002579	2011-05-12	17
6	002579	2011-05-13	18
7	002579	2011-05-16	18
8	002579	2011-05-17	18
9	002579	2011-05-18	17

✦ 6.5 Multilevel grouping & aggregation



SPL code

	A
1	=file("D:/STOCK.ctx").create().cursor().fetch()
2	=A1.group@u(SCODE)
3	=A2.(~.sort(TDAY))
4	=A3.(~.group@i((EPRICE<=EPRICE[-1]):GROUP;SCODE,(~.count()-1):COUNT))
5	=A4.(~.maxp(COUNT))

A2: Group data by stock code

A3: Sort data within a group by date

A4: Create a new group and count the days when the closing price does not rise

A5 Get the subgroup in each group with the largest count

SPL output

Index	GROUP	SCODE	COUNT
1	true	002579	2
2	true	002580	2
3	true	002581	2
4	true	002582	2
5	true	002583	2
6	true	002584	2
7	true	002585	2
8	true	002586	2
9	true	002587	2

✦ 6.6 Ordered grouping of big data



With a large amount of data, when the field value of the next record changes, put it to a new group and perform an aggregation.

Below is a huge log file where logs are output in the order of date and time. The task is to find the day having the most consecutive ERROR level.

Date	Time	Level	IP	...
2020/1/1	0:00:01	INFO	166.253.153.234	...
2020/1/1	0:00:02	INFO	99.72.133.239	...
2020/1/1	0:00:04	WARN	99.11.105.39	...
2020/1/1	0:00:05	INFO	117.69.80.195	...
2020/1/1	0:00:11	INFO	79.195.137.228	...
...

✦ 6.6 Ordered grouping of big data



SPL script is as follows, where `cs.group()` function creates a new group whenever the next value of the grouping field is different:

	A	B
1	<code>=file("ServerLog.txt").cursor@t()</code>	/Create cursor of the log file
2	<code>=A1.group(Date,Level;count(~):Count)</code>	/cs.group() creates a new group whenever the next date or log level is different
3	<code>=A2.select(Level:"ERROR")</code>	/Get groups of ERROR level
4	<code>=A3.top(1;ErrorCount)</code>	/Find the group that has the most consecutive ERRORS

A4	Date	ErrorCount
	2020/01/02	4

✦ 6.7 Ordered conditional grouping of big data



With a large amount of data, loop through records to calculate the expression and create a new group if the expression result is true.

Below is a huge log file where logs are output in the order of date and time. The task is to find the day having the most consecutive ERROR level.

Date	Time	Level	IP	...
2020/1/1	0:00:01	INFO	166.253.153.234	...
2020/1/1	0:00:02	INFO	99.72.133.239	...
2020/1/1	0:00:04	WARN	99.11.105.39	...
2020/1/1	0:00:05	INFO	117.69.80.195	...
2020/1/1	0:00:11	INFO	79.195.137.228	...
...

✦ 6.7 Ordered conditional grouping of big data



SPL script is as follows, where `cs.group()` function works with `@i` option to create a new group whenever the condition is changed:

	A	B
1	<code>=file("ServerLog.txt").cursor@t()</code>	/Create cursor of the log file
2	<code>=A1.group@i(Date[-1] != Date Level[-1] != Level;Date,Level,count(~):Count)</code>	/cs.group() uses @i option to create a new group whenever the condition is different
3	<code>=A2.select(Level:"ERROR")</code>	/Get groups of ERROR level
4	<code>=A3.top(1;ErrorCount)</code>	/Find the group that has the most consecutive ERRORS

A4	Date	ErrorCount
	2020/01/02	4



Chapter 7

SPL COOKBOOK

Loop calculation

✦ 7.1 Add new members to a sequence in loop



Loop through records to make judgment, adding new member at the end of a specified sequence each time.

Compare how many rows of data are identical in two files with the same number of rows.

ID	Predicted_Y	Original_Y
10	0.012388464367608093	0.0
11	0.01519899123978988	0.0
13	0.0007920238885061248	0.0
19	0.0012656367468159102	0.0
21	0.009460545997473379	0.0
23	0.024176791871681664	0.0
...

✦ 7.1 Add new members to a sequence in loop



SPL script is as follows, where "|" is used to concatenate a sequence and single values together:

	A	B	C
1	=file("p_old.csv").import@ct()		/Read the first output file
2	=file("p_new.csv").import@ct()		/Read the second output file
3	for A1.len()	=cmp(A1(A3),A2(A3))	/Iteratively compare records at the same position from two files
4		=@ B3	/Combine the result of each comparison with B4's value
5	=B4.count(~==0)		/Count how many rows are equal

B4	Member
	0
	0
	0
	...

A5	Value
	11302

✦ 7.2 Loop assignment



Calculate members of a sequence circularly.

Based on the sales table, give the salespersons whose results rank top 10% in 2014 5% performance awards.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 7.2 Loop assignment



SPL script is as follows, where A.run() function is used to assign the values of sequence members circularly.

	A	B
1	=connect("db").query("select * from sales")	/Connect data source and read the sales table
2	=A1.select(year(OrderDate)==2014)	/Select records of 2014
3	=A2.groups(SellerId;sum(Amount):Amount)	/Use groups function to group records by salesperson and sum the total sales of the current year
4	=A3.sort@z(Amount).to(A3.len()*0.1)	/Sort in descending order by sales amount and get records of the top 10%
5	=A4.run(Amount*=1.05)	/Loop through records of the top 10% to give each 5% performance awards

A5	SellerId	Amount
	4	150433.185
	3	127878.04
	1	102756.759
	8	87965.346

✦ 7.3 Loop calculation: complex inter-row calculation



Group records and loop each group to perform an aggregate over members of the current group over the target column while performing an inter-row calculation.

Part of the payment table is as follows:

ID	customID	name	amount_payable	due_date	amount_paid	pay_date
112101	C013	CA	12800	2014-02-21	12800	2014-12-19
112102	C013	CA	3500	2014-06-15	3500	2014-12-15
112103	C013	CA	2600	2015-03-21	6900	2015-10-17

According to the specified year (such as 2014), output the monthly payable amount. If there is no data of the current month, the payable amount is the value of the previous month:

name	1	2	3	4	5	6	7	8	9	10	11	12
CA		12800	12800	12800	12800	3500	3500	3500	3500	3500	3500	3500
...												

✦ 7.3 Loop calculation: complex inter-row calculation



SPL script is as follows, where there is a loop calculation within the current member:

	A	B
1	\$select * from Payment.txt where year(due_date)=2014	/Import payment data in 2014
2	=create(name,\${12.concat@c()})	/Create a table sequence table according to the target structure
3	=A1.group(customID).((m12=12.(null),~.(m12(month(due_date))=amount_payable), m12.(~=ifn(~,~[-1])),A2.record(name m12)))	/Group by customer ID, loop through each group to perform loop calculation over members in the current group and calculate the payable amount per month. Then insert results to A2's table sequence together with the customer names

A3												
name	1	2	3	4	5	6	7	8	9	10	11	12
CA		12800	12800	12800	12800	3500	3500	3500	3500	3500	3500	3500
NK			5800	5800	5800	9600	9600	9600	9600	3100	3100	3100
...												

✦ 7.4 Loop calculation: maximum continuous rising days



Within a loop calculation, calculate the number of continuous rising of values in a specified column. According to the records of SSE Composite Index, find the maximum number of days when the closing prices increase continuously in 2019.

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
2019/12/25	2980.4276	2981.8805	1.9E11
...

✦ 7.4 Loop calculation: maximum continuous rising days



SPL script is as follows, where ~ is used to represent the current member in the loop:

	A	B
1	=file("000001.csv").import@ct()	/Import data file
2	=A1.select(year(Date)==2019).sort(Date)	/Select records of 2019 and sort them by date
3	=n=0,A2.max(if(Close>Close[-1],n+=1,n=0))	/Loop through closing prices to compare the current closing price with that of the previous day. If the current closing price is higher, add 1 to the count, and finally select the maximum count value

A3	Value
	6

✦ 7.5 Loop calculation: nested loop



Use a nested loop function to calculate.

Here's the Hundred Fowls Problem. Now one cock is worth 5 qian, one hen 3 qian and 3 chicks 1 qian. It is required to buy 100 fowls with 100 qian. In each case, find the number of cocks, hens and chicks bought.

✦ 7.5 Loop calculation: nested loop



SPL script is as follows, where ~ is used to represent the current member in the loop:

	A	B
1	=to(100/5)	/Possible number of cocks to be purchased
2	=to(100/3)	/Possible number of hens to be purchased
3	=33.(~*3)	/Possible number of chicks to be purchased
4	=create(Cock,Hen,Chick)	/Create a table sequence to store the numbers of cocks, hens and chicks
5	>A1.run(A2.run(A3.run(if(A1.~+A2.~+A3.~==100 && A1.~*5+A2.~*3+A3.~/3==100,A4.insert(0,A1.~,A2.~,A3.~))))))	/Loop cock, hen, chick respectively, and when the condition is met, insert the result to A4's table sequence. The ~ symbol is used to represent the current member of looping through the sequence

A5	Cock	Hen	Chick
	4	18	78
	8	11	81
	12	4	84

✦ 7.6 Loop calculation: loop number



Search a text file by loop to the desired results, during which the loop number is used.

According to the keywords in text 1 to search text 2, and output according to the target format.

file1	file2	Output
like parks	I like to go out because I like parks.	Q1. like parks
went out	Ben does not go out much.	I
go out	Shelly went out often but does not like parks.	Shelly
	Harry does not go out neither does he like parks.	Harry
		Q2. went out
		Shelly
		Q3. go out
		I
		Ben
		Harry

✦ 7.6 Loop calculation: loop number



SPL script is as follows, where the # symbol is used to represent the current sequence number in the loop :

	A	B
1	=file("file1.txt").read@n()	/Read file1
2	=file("file2.txt").read@n()	/Read file2
3	=A1.conj(("Q"+string(#)+". "+~) A2.select(pos(~,A1.~)).(~.words()(1)))	/Loop through each string in file 1 to find it in file2 and get the first word of each matching string in file2. A2.select uses ~ to represent the current member of A2; A1.~ represents the current member of A1. Precede each group of searching result with "Q+ sequence number of A1's string +A1's current member", where the sequence number is got through #

A3	Member
	Q1. like parks
	I
	Shelly
	Harry
	Q2. went out
	Shelly
	Q3. go out
	I
	Ben
	Harry

✦ 7.7 Loop calculation: calculate adjacent data by position during the loop calculation



Calculate the average of values in an deviated interval by position during the loop calculation.

According to the trading table of China Merchants Bank, list the 20-day average closing price for each day from January 1 to 10, 2020.

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
2019/12/25	2980.4276	2981.8805	1.9E11
...

✦ 7.7 Loop calculation:calculate adjacent data by position during the loop calculation



SPL script is as follows, where A.calc() function calculates values and returns result according to the specified positions; [a:b] is used to access members in loop:

	A	B
1	=connect("db")	/Connect data source
2	=A1.query("select Date, Close from Stock where Code='600036' order by Date")	/Select records of China Merchants Bank and sort them by date
3	=A2.pselect@a(Date>=date("2020/01/01") && Date<=date("2020/01/10"))	/Use pselect() function to get sequence numbers of the records from January 1 to 10, 2020
4	=A2(A3).derive(A2.calc(A3(#),avg(Close[-19:0])):ma20)	/calc() function iteratively calculates and returns the 20-day average for each the first ten days. Expression Close[- 19:0] gets the closing prices from 19 days before to the current day

A3	Member
	4311
	4312
	4313
	4314
	4315
	4316
	4317

A4	Date	Close	ma20
	2020/01/02	38.88	37.35
	2020/01/03	39.4	37.50
	2020/01/06	39.24	37.64
	2020/01/07	39.15	37.79
	2020/01/08	38.41	37.90
	2020/01/09	38.9	38.03
	2020/01/10	39.04	38.16

✦ 7.8 Loop calculation: iterative accumulation



Iteratively accumulate during looping, and perform a filter over the accumulated values.
According to the sales table, calculate the number of days needed to achieve 20 orders per month in 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 7.8 Loop calculation: iterative accumulation



SPL script is as follows, where seq() function is used to generate sequence numbers:

	A	B
1	=connect("db").query("select * from sales")	/Connect the data source and read the sales table
2	=A1.select(year(OrderDate)==2014)	/Select records of 2014
3	=A2.sort(OrderDate)	/Sort by order date
4	=A3.select(seq(month(OrderDate))=20)	/Use seq() function to get sequence numbers of orders per month and select the record with sequence number being 20 in each month
5	=A4.new(month(OrderDate):Month,day(OrderDate):Day)	/List the number of days needed to reach 20 sales per month

A5	Month	Day
	1	20
	2	20
	3	20
	4	18

✦ 7.9 Loop calculation:group and calculate ranking



Iteratively calculate ranking in each group.

According to the employee income table, get the income ranking of each employee in the department.

ID	NAME	DEPT	SALARY
1	Rebecca	R&D	7000
2	Ashley	Finance	11000
3	Rachel	Sales	9000
4	Emily	HR	7000
5	Ashley	R&D	16000
...

✦ 7.9 Loop calculation:group and calculate ranking



SPL script is as follows, where rank() function is used to number members with a same field value:

	A	B
1	=connect("db") .query("select * from Employee order by DEPT, SALARY DESC")	/Connect data source, read employee table and sort it by department and salary
2	=A1.derive(rank(SALARY;DEPT):DeptRank)	/Use rank() function to number members with ordered department and salary and get ranking in each department

A2	ID	NAME	DEPT	SALARY	DeptRank
	2	Ashley	Finance	11000	1
	32	Andrew	Finance	11000	1
	230	Hannah	Finance	10000	3
	24	Chloe	Finance	10000	3

✦ 7.10 Loop calculation: calculate dense ranking in each group



Iteratively calculate the dense ranking for each group.

According to the score table, find the rankings of students with ID 8 in each subject in class one.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 7.10 Loop calculation: calculate dense ranking in each group



The ranki() function is used to number members with a same field value. The difference between ranki function and rank function is that the former returns consecutive numbers, which is similar to DENSE_RANK. SPL script is as follows:

	A	B
1	=connect("db") .query("select * from SCORES where CLASS='Class one' order by SUBJECT, SCORE DESC")	/Connect data source, read scores table and sort it by subject and score
2	=A1.derive(ranki(SCORE;SUBJECT):Rank)	/use ranki () function to number records with ordered subjects and scores, and calculate the dense ranking for each subject
3	=A2.select(STUDENTID==8)	/Select records of students with ID 8
4	=create(\${A3.(SUBJECT).concat@c()}).record(A3.(Rank))	/Get the dense ranking of each subject for each student in A3

A4	English	Math	PE
	10	4	14

✦ 7.11 Loop calculation: iterative sum



Calculate the result of iterative sum by loop.

According to the SSE Composite Index table, calculate the annual cumulative transaction amount of each trading day in 2019.

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
2019/12/25	2980.4276	2981.8805	1.9E11
...

✦ 7.11 Loop calculation: iterative sum



SPL script is as follows, where cum() function is used to calculate the cumulative transaction amount:

	A	B
1	=file("000001.csv").import@ct()	/Import data file
2	=A1.select(year(Date)==2019).sort(Date)	/Select records of 2019 and sort them by date
3	=A2.derive(cum(Amount):CUM)	/Use cum() function to calculate the cumulative transaction amount

A3	Date	Open	Close	Amount	CUM
	2019/01/02	2497.8805	2465.291	9.759E10	9.759E10
	2019/01/03	2461.7829	2464.3628	1.07E11	2.046E11
	2019/01/04	2446.0193	2514.8682	1.39E11	3.436E11
	2019/01/07	2528.6987	2533.0887	1.46E11	4.896E11
	2019/01/08	2530.3001	2526.4622	1.23E11	6.126E11

✦ 7.12 Loop calculation: custom iterative calculation



Perform an iteration calculation by defining the expression and the termination condition for the iterative process.

According to the sales table, find the day when the sales target of \$150000 is achieved in the first quarter of 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 7.12 Loop calculation: custom iterative calculation



SPL script is as follows, where A.iterate() function is used to perform iterative calculation over members:

	A	B
1	=connect("db").query("select * from sales")	/Connect the data source and read the sales table
2	=A1.select(year(OrderDate)==2014)	/Select records of 2014
3	=A2.iterate((@ +=Amount, ~~=OrderDate),0,@ > 150000)	/iterate() function performs an iterative calculation with an initial value of 0. Add up the sales amounts to the current cell value until the total is over 150000. The function returns an order date

A3	Value
	2014/3/25



Chapter 8

SPL COOKBOOK

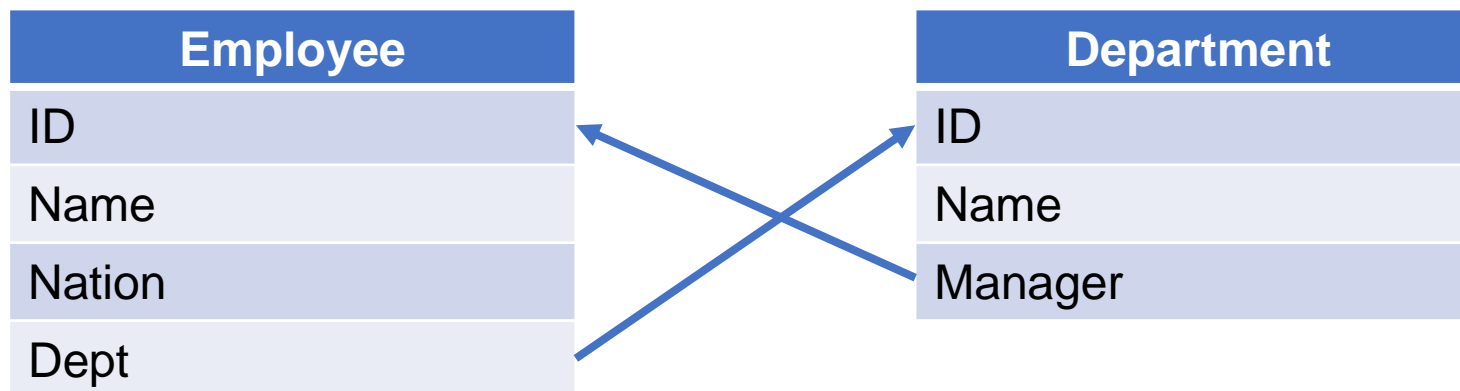
Join query over multiple tables

✦ 8.1 Perform filtering through multi-level association



Perform a multi-level associative query between two tables and then filtering.

Based on the associated *Employee* table and *Department* table, find which American employees have a Chinese manager.



✦ 8.1 Perform filtering through multi-level association



SPL script is as follows, where A.switch() function switches the foreign key field values to the corresponding records in the foreign key table:

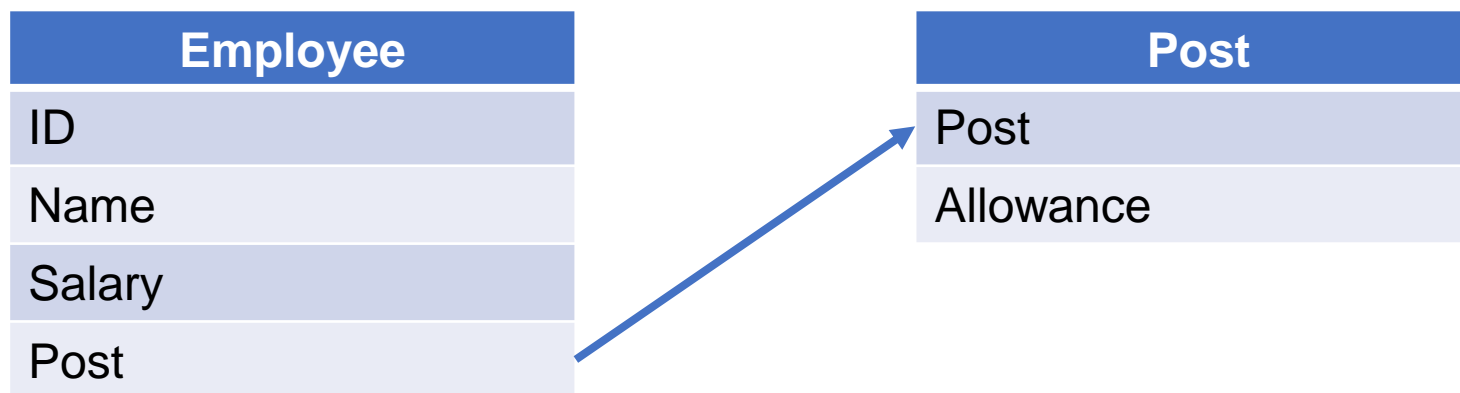
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Employee")	/Query <i>Employee</i> table
3	=A1.query("select * from Department")	/Query <i>Department</i> table
4	=A3.switch(Manager, A2:ID)	/Use switch function to switch the Manager filed in <i>Department</i> table to the corresponding records in <i>Employee</i> table
5	=A2.switch(Dept, A4:ID)	/Use switch function to switch the Dept field in <i>Employee</i> table to the corresponding records in <i>Department</i> table
6	=A5.select(Nation=="American" && Dept.Manager.Nation=="Chinese")	/Select employees whose nationality is America and whose manager's nationality is China

A6	ID	Name	Nation	Dept
	11	Simon	American	2
	103	Rudy	American	2

✦ 8.2 Switch foreign key field values to the corresponding records



Merge and calculate the associated data in two tables. The two tables may not match exactly. Based on the associated *Employee* table and *PostAllowance* table, calculate the total income of employees.



✦ 8.2 Switch foreign key field value to the corresponding record



SPL script is as follows, where A.switch() function switches the foreign key field values to the corresponding records in the foreign key table. If the corresponding record does not exist, set it as null:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Employee")	/Query <i>Employee</i> table
3	=A1.query("select * from PostAllowance")	/Query <i>PostAllowance</i> table
4	=A2.switch(Post, A3:Post)	/Use switch function to switch the post field in employee table to the corresponding records, and set to null if the corresponding record does not exist
5	=A4.new(ID,Name,Salary+Post.Allowance:Salary)	/Create a table sequence and calculate the total income

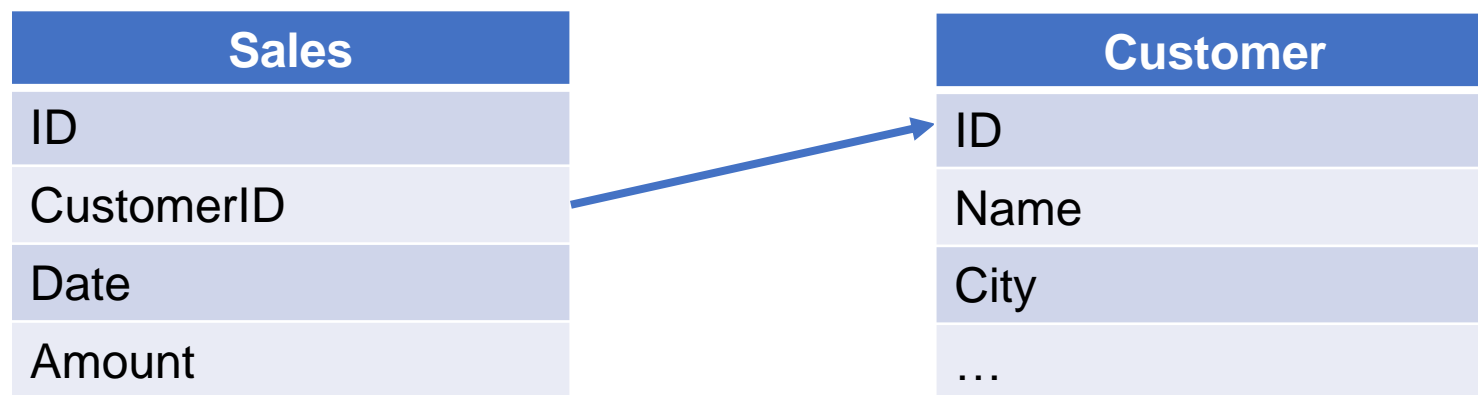
A5	ID	Name	Salary
	1	Rebecca	8000
	2	Ashley	12000

✦ 8.3 Get records by matched foreign key values



Between two associated tables, search for records according to the foreign key values that can be matched and then perform grouping and aggregation.

According to the associated *sales* table and *customer* table, find the total sales amount of each customer in Beijing in 2014.



✦ 8.3 Get records by matched foreign key values



@i option is used with the A.switch() function to delete the mismatched records. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Sales where year(Date)=2014")	/Query the data of sales table in 2014
3	=A1.query("select * from Customer where City='Beijing'")	/Query customers in Beijing
4	=A2.switch@i(CustomerID, A3:ID)	/Use @i option with switch function to keep only the records of customers in Beijing
5	=A4.groups(CustomerID.Name:Name; sum(Amount):Amount).sort@z(Amount)	/Group and sum each customer's sales amount, and sort them in descending order

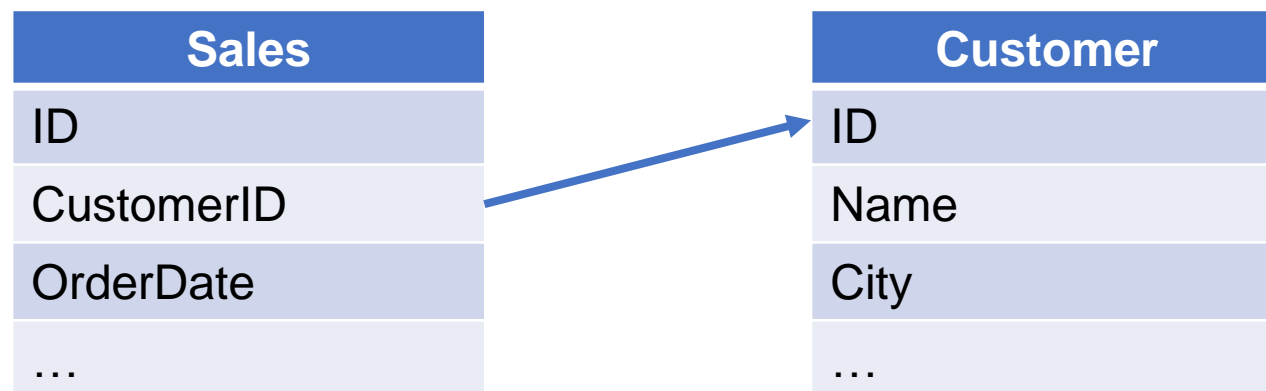
A5	Name	Amount
	SAVEA	130672.64
	HUN	23959.05

✦ 8.4 Get records by mismatched foreign key values



Between two associated tables, search for records according to the foreign key values that cannot find matches.

Query the new customers in 2014 based on the associated *Sales* table and *Customer* table.



✦ 8.4 Get records by mismatched foreign key values



SPL script is as follows, where @d option works with A.switch() function to get only the mismatched records (the foreign key field will not be set to null in this case):

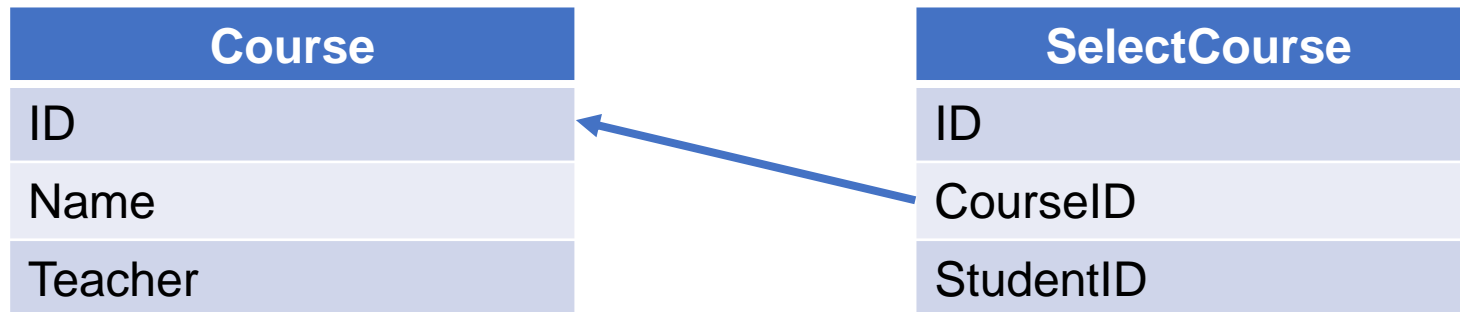
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Sales where year(OrderDate)=2014")	/Query sales records in 2014
3	=A1.query("select * from Customer")	/Query customer table
4	=A2.switch@d(CustomerID ,A3:ID)	/use switch@d() to select the records from the sales table that the customer ID does not exist in the customer table

A4	ID	CustomerID	OrderDate	...
	10439	MEREP	2014/02/07	...
	10504	WHITC	2014/04/11	...

✦ 8.5 Join query over two tables



Between two associated tables, filter records according to the join condition and do calculation.
According to the *Course* table and *SelectCourse* table, find how many students there are who have selected the "Matlab" course.



✦ 8.5 Join query over two tables



@i option is used with the A.join() function to delete the mismatched records. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Course")	/Query <i>Course</i> table
3	=A1.query("select * from SelectCourse")	/Query <i>SelectCourse</i> table
4	=A2.select(Name=="Matlab")	/Select the specified course from the <i>Course</i> table
5	=A3.join@i(CourseID,A4:ID).count()	/Use @i option in the join function to perform a join filter, and then count

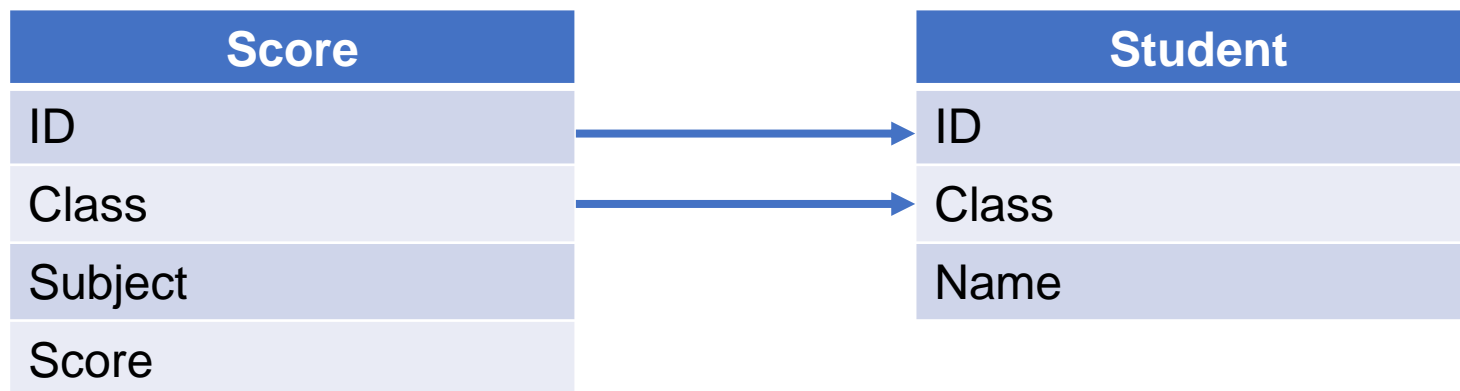
A5	Value
	5

✦ 8.6 Perform a multi-field join and conditional filtering over two tables



Between two associated tables, filter records according to the multi-field join condition, and then group & aggregate data.

Based on the associated *Score* table and *Student* table, calculate the total score of each student in class one.



✦ 8.6 Perform a multi-field join and conditional filtering over two tables



@i option is used with the A.join() function to delete the mismatched records. During the join, specify the class as a constant condition (Class one) to do the join filter. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Score")	/Query the <i>Score</i> table
3	=A1.query("select * from Student")	/Query the <i>Student</i> table
4	=A2.join@i(ID:"Class one", A3:ID:Class)	/Use A.join@i() function to filter with ID when performing a multi-field join by ID and Class
5	=A4.groups(ID; sum(Score):TotalScore)	/Group and calculate the total score of each student

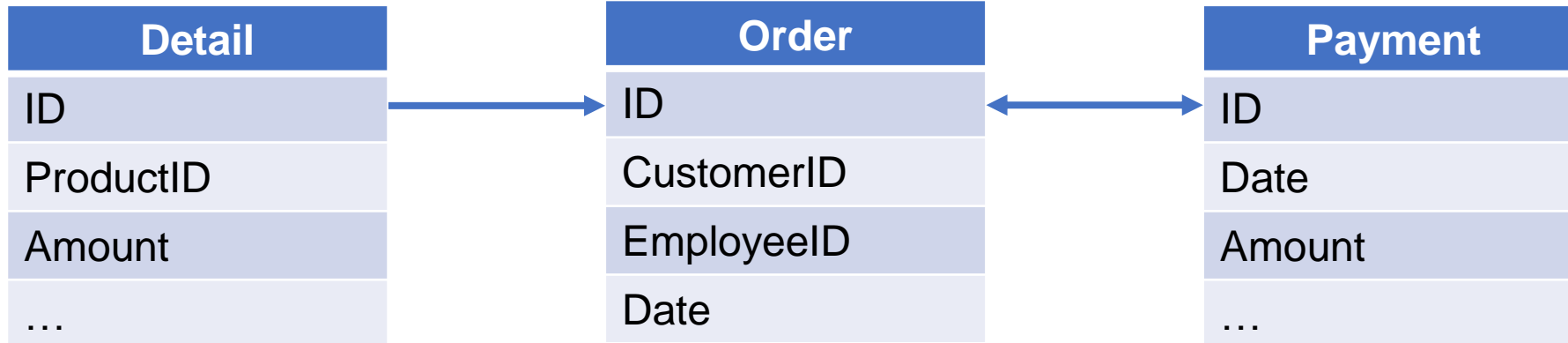
A5	ID	TotalScore
	1	230
	2	258
	3	228

✦ 8.7 Join query over multiple tables



Among three associated tables, filter records according to the join condition and do calculation.

Based on associated *Order* table, *Detail* table and *Payment* table, query which orders have not been fully paid.



✦ 8.7 Join query over multiple tables



join() function is used to perform a join. SPL script is as follows:

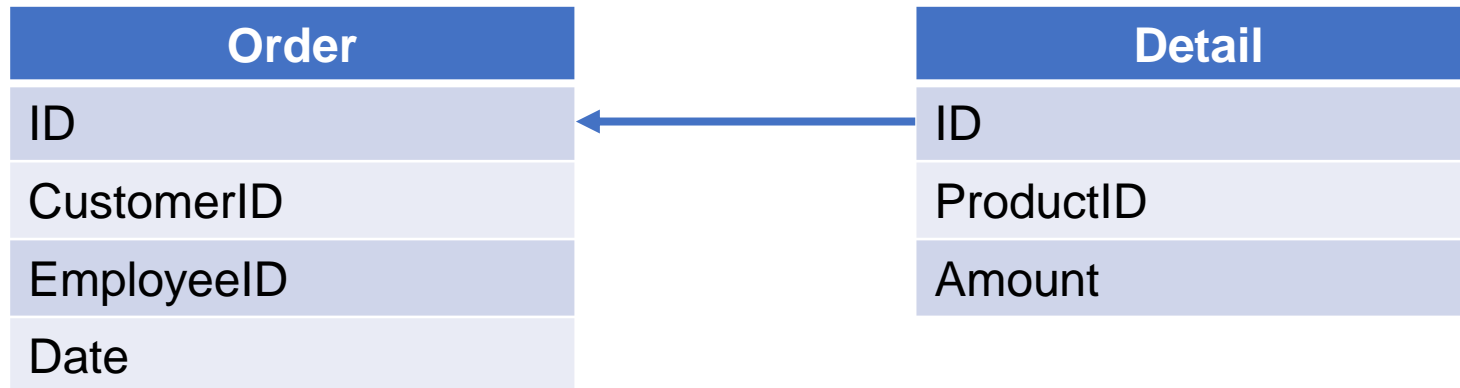
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Order")	/Query <i>Order</i> table
3	=A1.query("select * from Detail")	/Query <i>Detail</i> table
4	=A1.query("select * from Payment")	/Query <i>Payment</i> table
5	=A3.group(ID)	/Group <i>Detail</i> table by order ID
6	=A4.group(ID)	/Group <i>Payment</i> table by order ID
7	=join(A2:Order,ID; A5:Detail,ID; A6:Payment,ID)	/Use join function to join <i>Order</i> table, <i>Detail</i> table and <i>Payment</i> table by order ID
8	=A7.new(Order.ID:ID,Detail.sum(Amount):Amount,Payment.sum(Amount):Pay)	/Create a table sequence, get the sum of each order and payment
9	=A8.select(Pay<Amount)	/Select the records whose payment amount is less than the order amount

A9	ID	Amount	Pay
	AROUT	55492.0	35980
	BERGS	3398.55	1080

✦ 8.8 Join two tables of the same order by merging



Join two tables of the same order by merging, then group & aggregate the result.
Based on the associated *order* table and *detail* table, calculate the sales amount of each customer in 2014.



✦ 8.8 Join two tables of the same order by merging



@m option is used with join() function to perform the order-based merge. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Order where year(Date)=2014 order by ID")	/Query orders in 2014, and sort them by order ID
3	=A1.query("select * from Detail order by ID")	/Query <i>Detail</i> table and sort it by order ID
4	=join@m(A2:Order,ID;A3:Detail,ID)	/Use join@m function to merge <i>Order</i> table and <i>Detail</i> table in order
5	=A4.groups(Order.CustomerID:CustomerID; sum(Detail.Amount):Amount)	/Group and aggregate the sales amount of each customer

A5	CustomerID	Amount
	ALFKI	14848.0
	ANTON	4041.0

✦ 8.9 Join big data tables of the same order by merging



There are multiple associated tables that are ordered, including big data tables. Perform an order-based merge and filter records.

Based on the associated *Order* table, *Detail* table and *Customer* table, find the customers whose total sales amount exceeds 10000. The *Detail* and *Order* tables have large amounts of data that cannot be fully loaded into the memory.



✦ 8.9 Join big data tables of the same order by merging



joinx() function is used here to perform order-based merge. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.cursor("select * from Order order by ID")	/Create a cursor for <i>Order</i> table
3	=A1.cursor("select * from Detail order by ID")	/Create a cursor for <i>Detail</i> table
4	=A1.query("select * from Customer")	/Query <i>Customer</i> table
5	=A2.switch@i(CustomerID,A4:ID)	/Use switch@i function to switch CustomerID to the corresponding records in <i>Order</i> table and delete non-matched records
6	=joinx(A5:Order,ID;A3:Detail,ID)	/Use joinx function to merge the cursors of <i>Order</i> table and <i>Detail</i> table in order
7	=A6.groups(Order.CustomerID.Name;sum(Detail.Amount):Amount).select(Amount>10000)	/Group and aggregate the sales amount of each customer, and select records with sales amount more than 10000

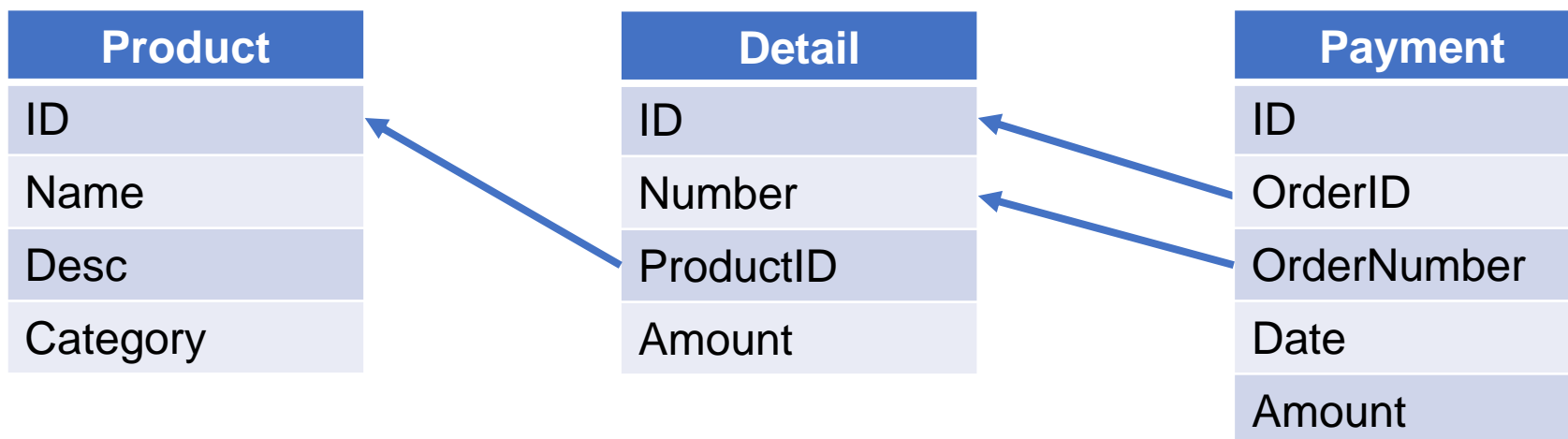
A7	Name	Amount
	ALFKI	14848.0
	AROUT	55492.0

✦ 8.10 Perform a left join by multi-field primary key of dimension table



Perform a left join over associated tables by multi-field primary key to filter records.

Based on the associated *order* table, *details* and *payment* table, find the products with payment record in 2014 and single order amount exceeding 500.



✦ 8.10 Perform a left join by multi-field primary key of dimension table



A.join() function is used here to perform a left join by multi-field primary key. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Detail")	/Query <i>Detail</i> table
3	=A1.query("select * from Payment")	/Query <i>Payment</i> table
4	=A1.query("select * from Product")	/Query <i>Product</i> table
5	=A2.switch@i(ProductID,A4:ID)	/Use switch@i function to switch ProductID in <i>Detail</i> table to the corresponding record
6	=A3.join(OrderID:OrderNumber,A5:ID:Number,~:Detail)	/Use A.join function to join <i>Detail</i> table and <i>Payment</i> table
7	=A6.select(year(Date)==2014 && Detail.Amount>500)	/Select the payment records in 2014 with order amount over 500
8	=A7.new(ID,Date,Detail.Product.Name:Name,Detail.Amount:Amount)	/Use selected results to create a table sequence

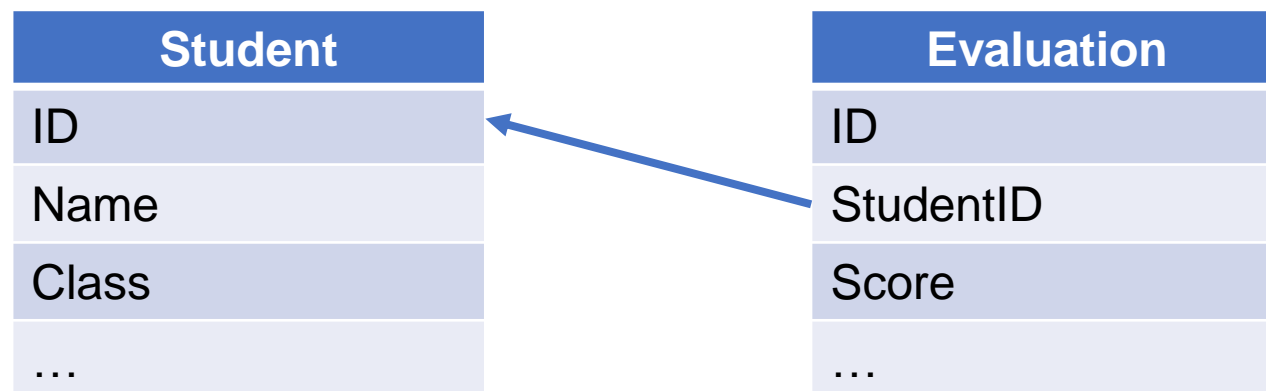
A8	ID	Date	Name	Amount
	10979	2014/03/26	Soda water	1317
	11011	2014/04/09	Espresso	530

✦ 8.11 Perform a left join between two tables



Perform cross-table calculation by left join of two tables.

Based on the associated *Student* table and *Evaluation* table, find the score of each student. The basic score of each student is 70, and then adjusted according to the evaluation score.



✦ 8.11 Perform a left join between two tables



@1 option is used with join() function to perform a left join. Based on the left table sequence, null is used if there is no matching records in the right table. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Students")	/Query <i>Students</i> table
3	=A1.query("select * from Evaluation")	/Query <i>Evaluation</i> table
4	=A3.group(StudentID)	/Group <i>Evaluation</i> table by StudentID
5	=join@1(A2:Students,ID;A4:Evaluation,StudentID)	/Use join@1() function to perform a left join between <i>Students</i> table and grouped <i>Evaluation</i> table
6	=A5.new(Students.ID:ID,Students.Name:Name,70+Evaluation.sum(Score):Score)	/Create a table sequence, and calculate the total score of each student (70+Evaluation score)

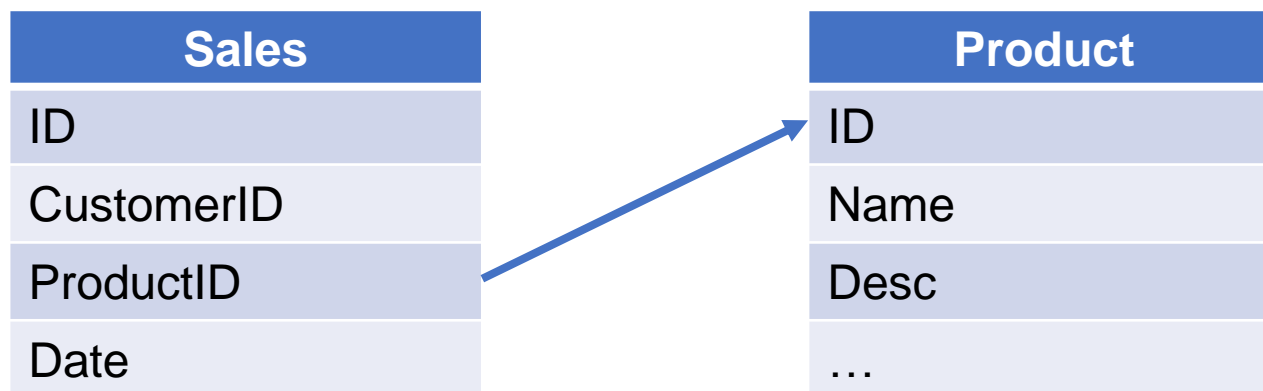
A6	ID	Name	Score
	1	Ashley	85
	2	Rachel	65
	3	Emily	70

✦ 8.12 Perform a full join between two tables



Perform cross-table calculation by full join of two tables.

Based on the associated *sales* table and *product* table, find which products are sold in each month of 2014.



✦ 8.12 Perform a full join between two tables



SPL script is as follows, where @f option is used with join() function to perform a full join:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select ProductID, month(Date) as Month from Sales where year(Date)=2014")	/Query the sales records of 2014
3	=A1.query("select * from Product")	/Query <i>Product</i> table
4	=A2.switch(ProductID ,A3:ID)	/Use switch function to switch ProductID to the corresponding records
5	=A4.group(Month)	/Group and sort by month
6	=A5.(~.group@1(ProductID).new(ProductID.Name:Product, count(~):Count))	/Deduplicate by product for each month, and get the product name field
7	=A6.("A6(" + string(#) + "):" + string(#) + ",Product").concat(";")	/Concatenate the parameter string of join@f() function
8	=join@f(\$A7)	/Use join@f() function to perform a full join over 12 months' data

A8											
1	2	3	4	5	6	7	8	9	10	11	12
(null)	[Cheese,3]	(null)	(null)	(null)	(null)	(null)	(null)	(null)	[Cheese,6]	(null)	(null)
(null)	[Coffee,7]	(null)	[Coffee,6]	[Coffee,9]	(null)	[Coffee,9]	(null)	(null)	(null)	(null)	[Coffee,8]
[Milk,3]	(null)	[Milk,5]	[Milk,7]	(null)	[Milk,6]	[Milk,8]	[Milk,3]	(null)	[Milk,6]	[Milk,4]	(null)
...

✦ 8.13 Cartesian product with filter condition



Get the Cartesian product of two associated tables and then perform filtering.

According to the *Sandwich* table and *Ingredient* table, find which two sandwiches use the most similar ingredients.

Sandwich		
ID	Name	Price
1	BLT	5.5
2	Reuben	7.0
3	Grilled Cheese	3.75

Ingredient	
ID	Ingredient
1	bacon
1	lettuce
1	tomato
...	...

✦ 8.13 Cartesian product with filter condition



xjoin() function is used here to calculate the cross product. The SPL cross product result is composed of records of two tables, rather than simply expanding all fields. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select i.ID ID, i.Ingredient Ingredient, s.Name Name from Sandwich s, Ingredient i where s.ID=i.ID order by ID")	/Query <i>Sandwich</i> table and <i>Ingredient</i> table
3	=A2.group@o(ID;Name,~.(Ingredient):Collection)	Use group@o() to group by ID, and store the ingredients of each sandwich to Collection field
4	=xjoin(A3:A;A3:B,A.ID<ID)	/Use xjoin function to calculate cross product, and select the combinations that IDs are different
5	=A4.new((A.Collection ^ B.Collection).len():Count, A.Name:Name1, B.Name:Name2).sort@z(Count)	/Calculate the number of duplicate ingredients in the two formulations and arrange them in descending order

A5	Count	Name1	Name2
	1	Reuben	Grilled Cheese
	0	BLT	Reuben
	0	BLT	Grilled Cheese

✦ 8.14 Use Cartesian product to calculate matrix multiplication



Use Cartesian product to calculate matrix multiplication.

Matrix
row
col
value

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

The mathematical formula of this example is as follows:

$$C = AB = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 1 \times 1 + 2 \times 2 + 3 \times 3 & 1 \times 4 + 2 \times 5 + 3 \times 6 \\ 4 \times 1 + 5 \times 2 + 6 \times 3 & 4 \times 4 + 5 \times 5 + 6 \times 6 \end{pmatrix} = \begin{pmatrix} 14 & 32 \\ 32 & 77 \end{pmatrix}$$

✦ 8.14 Use Cartesian product to calculate matrix multiplication



SPL script is as follows, where xjoin() function is used to calculate the cross product and perform the conditional filtering at the same time:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from MatrixA")	/Query matrix table A
3	=A1.query("select * from MatrixB")	/Query matrix table B
4	=xjoin(A2:A; A3:B, A.col==A3.row)	/Use xjoin function to calculate the cross product of two tables, and perform conditional filtering at the same time
5	=A4.groups(A.row:row,B.col:col;sum(A.value * B.value):value)	/Group and aggregate the results to calculate the values of each row and column

A5	row	col	value
	1	1	14
	1	2	32
	2	1	32
	2	2	77

✦ 8.15 Use left join to calculate Cartesian product



Left join two tables to calculate Cartesian product.

Based on the associated *Community* table and *Age* table, find the age group each community resident belongs to.

Community		
ID	Name	Age
1	David	28
2	Daniel	15
3	Andrew	65
4	Rudy	

Age		
Group	Start	End
Children	0	15
Youth	16	40
Middle	41	60
Old	61	100

✦ 8.15 Use left join to calculate Cartesian product



SPL script is as follows, where @1 option is used with xjoin() function to calculate the cross product through the left join:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Community")	/Query <i>Community</i> table
3	=A1.query("select * from Age")	/Query <i>Age</i> table
4	=xjoin@1(A2:Person; A3:Age, A3.Start<=Person.Age && A3.End>=Person.Age)	/Use xjoin@1() function to calculate cross product by left join, and select the records of age in the corresponding age range
5	=A4.new(Person.ID:ID, Person.Name:Name, Person.Age:Age, Age.Group:Group)	/Create a table sequence to return the age group of each person

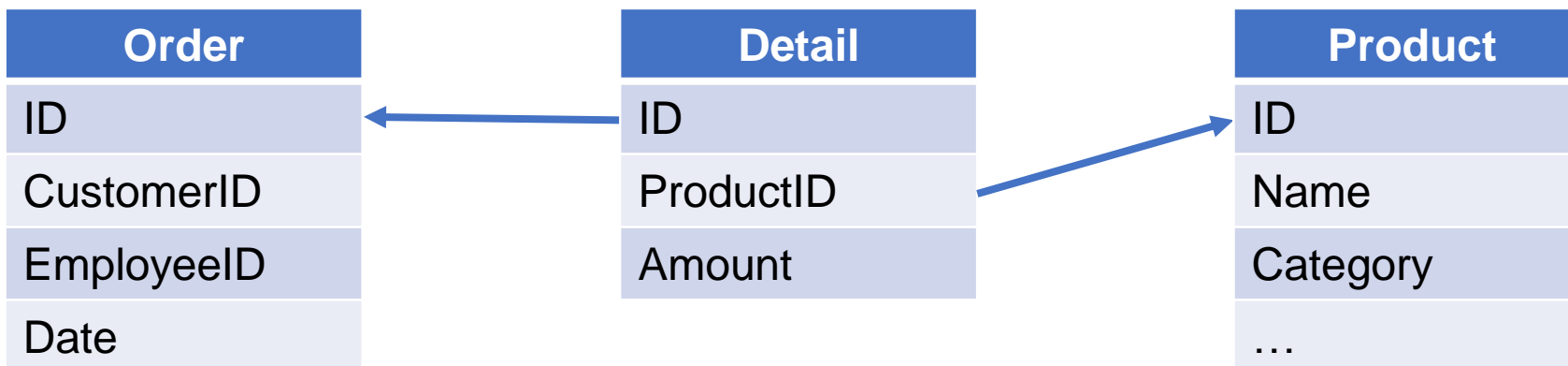
A5	ID	Name	Age	Group
	1	David	28	Youth
	2	Daniel	15	Children
	3	Andrew	65	Old
	4	Rudy	(null)	(null)

✦ 8.16 Join query between big data tables and large dimension table



Join query over two big data tables and an ordered btx file.

Based on the associated *Order* table, *Detail* and *Product* table, find the total sales quantity of each product in January 2014. The *Order* table and *Detail* table are big data files, and the *product* table is a btx file ordered by ID.



✦ 8.16 Join query between big data tables and large dimension table



cs.joinx() function is used here to join files, where the btx file should be ordered by the joining field. SPL script is as follows:

	A	B
1	=file("Detail.ctx").create().cursor()	/Create a cursor for <i>Detail</i> table
2	=file("Order.ctx").create().cursor(;year(Date)=2014 && month(Date)=1)	/Create a cursor for <i>Order</i> table in Jan 2014
3	=file("Product.btx")	/Create <i>Product.btx</i> object
4	=A1.joinx@i(ID,A2:ID)	/Use @i option with cs.joinx function to perform a join filter
5	=A4.joinx(ProductID,A3:ID,Name:ProductName)	/Use cs.joinx function to join <i>Detail</i> table and <i>Product</i> table by ProductID
6	=A5.groups(ProductName; count(~):Count)	/Group and aggregate the sales quantity of each product

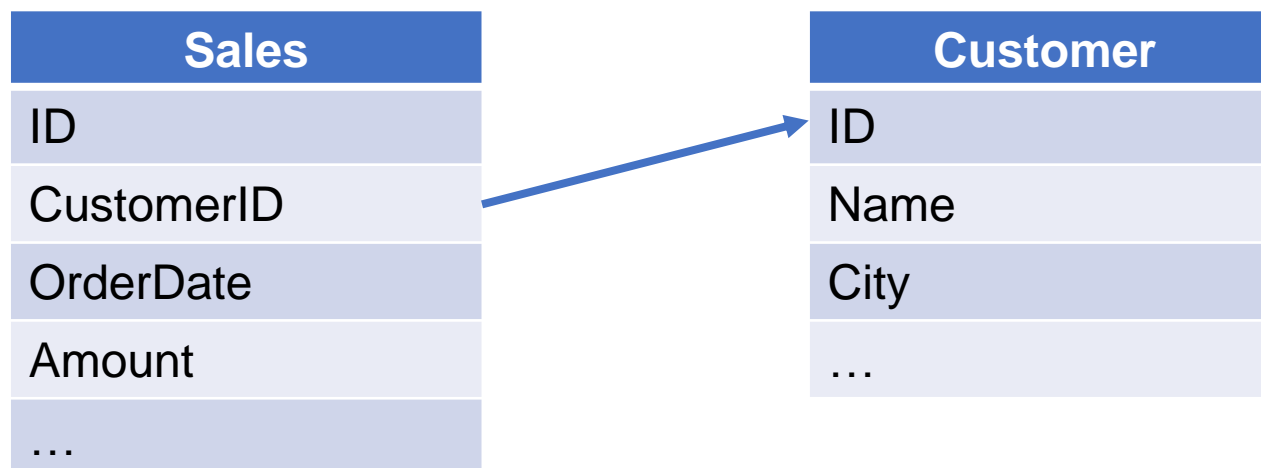
A6	ProductName	Count
	Milk	32
	Coffee	60

✦ 8.17 Fast join query between small data table and large dimension table



After a cursor is generated for a table with a small amount of data, perform a fast join query between it and an ordered btx file.

Based on the associated btx file *Sales* and btx file *Customer*, find the customer names with top three sales amounts in 2014.



✦ 8.17 Fast join query between small data table and large dimension table



Use `cs.joinx()` function to join with a segmentable btx file. When the amount of data is small, `@q` option can be used to speed up the join. SPL script is as follows:

	A	B
1	<code>=file("Sales.btx").cursor@b().select(year(Date)==2014)</code>	/Create a cursor for Sales.btx, and select records of 2014
2	<code>=file("Customer.btx")</code>	/Create Customer.btx object, which is ordered by CustomerID
3	<code>=A1.groups(CustomerID;sum(Amount):Amount)</code>	/Group by customerID and get sum of sales amount of each customer
4	<code>=A3.top(-3;Amount)</code>	/Select top3 sales amounts
5	<code>=A4.joinx@q(CustomerID,A2:ID,Name:CustomerName).fetch()</code>	/Use <code>cs.joinx</code> function to join <i>Sales</i> table and <i>Customer</i> table by CustomerID, and <code>@q</code> option is used to speed up the join as the data amount is small

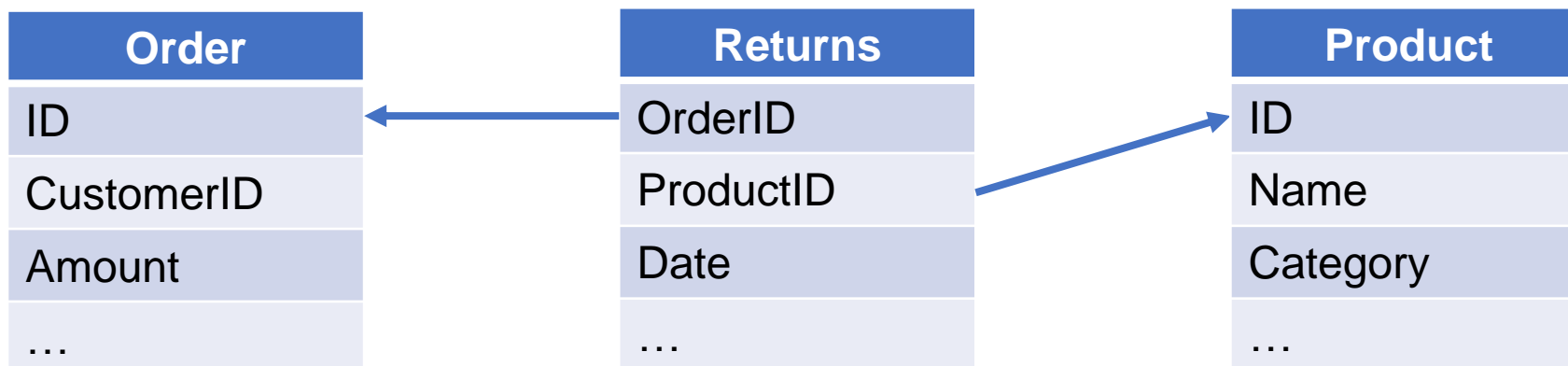
A5	CustomerID	Amount	CustomerName
	71	130672.64	SAVEA
	63	64238.0	QUICK
	20	53467.38	ERNSH

✦ 8.18 Fast join query over same-order data tables and large dimension table



The fast join query over two cursors and an ordered btx file. It requires that the cursor be ordered by the first join field.

Based on the associated *Order* table, *Returns* table and *Product* table, find the total refund of each product in 2015.





cs.joinx() function is used to join files. @c option can be used to speed up the join if the cursor is ordered by the first join field. You can also use @c and @q at the same time. SPL script is as follows:

	A	B
1	=file("Returns.btx").cursor@b().select(year(Date)==2015)	/Create a cursor for <i>Returns.btx</i>
2	=file("Order.btx")	/Create <i>Order.btx</i> file object
3	=file("Product.btx")	/Create <i>Product.btx</i> file object
4	=A1.joinx@qc(OrderID,A2:ID,Amount;ProductID,A3:ID,Category)	/Use cs.joinx function to join <i>Order.btx</i> by OrderID, and <i>Product.btx</i> by ProductID. @qc options are used together to speed up the join
5	=A4.groups(Category; sum(Amount))	/Group by category and sum the amount

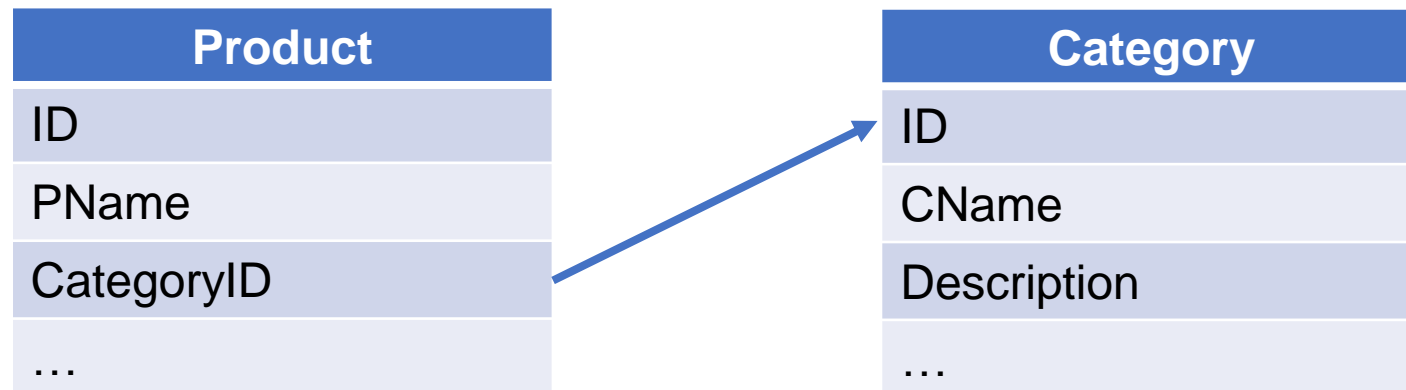
A5	Category	Amount
	Electric appliance	1854.5
	Fruits	251.5

✦ 8.19 Join two tables through locating records by sequence numbers



Join two tables by locating records according to their sequence numbers.

Based on the associated *Product* table and *Category* table, find all products with "drink" in their category name.



✦ 8.19 Join two tables through locating records by sequence numbers



A.join() function is used to perform a join, where # is used to locate a record by its sequence number. SPL script is as follows:

	A	B
1	=connect("demo")	/Connect to database
2	=A1.query("select * from Product")	/Query <i>Product</i> table
3	=A1.query("select * from Category")	/Query <i>Category</i> table
4	=A2.join(CategoryID,A3:#,CName)	/Use A.join() function to join <i>Category</i> table by CategoryID, which is located in <i>Category</i> table by sequence numbers, and add the foreign key field CName
5	=A4.select(like@c(CName, "*drink*"))	/Select records that contain the string "drink" in the category name; case insensitive

A5	ID	Name	CategoryID	CName
	24	Soda	1	Drink
	34	Beer	1	Drink
	35	Orange Juice	1	Drink

✦ 8.20 Perform an alignment join by positions to shuffle values of a field



To shuffle and encrypt the letters through an alignment join.

Shuffle and encrypt values of a certain column in the database and then write them back to the database.

ID	ORIGINAL_VALUE	SHUFFLED_VALUE
1	D	N
2	U	n
3	j	K
4	N	D
...

✦ 8.20 Perform an alignment join by positions to shuffle values of a field



@p option is used with join() function to perform an alignment join by sequence numbers. SPL script is as follows:

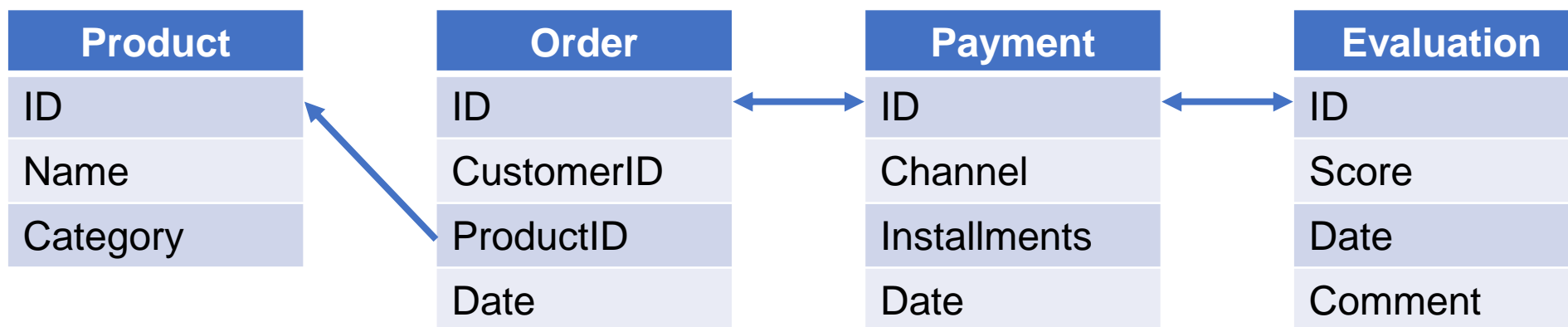
	A	B
1	=connect("demo")	/Connect to database
2	=A1.query("select ID,ORIGINAL_VALUE from REF_VALUES")	/Query <i>REF_VALUES</i> table
3	=A2.sort(rand())	/Sort by random seed to shuffle the table sequence
4	=join@p(A2.(ID);A3.(ORIGINAL_VALUE))	/Use join@p to join IDs and shuffled values in pair by sequence numbers
5	=A1.update@u(A4, REF_VALUES, ID:_1, SHUFFLED_VALUE:_2; ID)	/According to the primary key ID, update the result of A4 to the database table Ref_VALUES

✦ 8.21 Perform alignment join over multiple tables by sequence numbers



Perform alignment join over four associated tables according to sequence numbers, then group and calculate the average for each group.

Based on the associated *Order* table, *Payment* table, *Product* table and *Evaluation* table, find the average evaluation score of each category of products in the orders that did not use installment payment in 2014.



✦ 8.21 Perform alignment join over multiple tables by sequence numbers



@p option is used with join() function to perform a join by positions. SPL script is as follows:

	A	B
1	=connect("demo")	/Connect to database
2	=A1.query("select * from Order order by ID")	/Query <i>Order</i> table
3	=A1.query("select * from Payment order by ID")	/Query <i>Payment</i> table
4	=A1.query("select * from Evaluation order by ID")	/Query <i>Evaluation</i> table
5	=A1.query("select * from Product")	/Query <i>Product</i> table
6	=A2.switch(ProductID, A5:ID)	/Use switch function to switch ProductID to corresponding records in <i>Product</i> table
7	=join@p(A6:Order;A3:Payment;A4:Evaluation)	/Use join@p to join three tables by positions
8	=A7.select(year(Order.Date)==2014 && !Payment.Instalments)	/Select orders not using instalments in 2014
9	=A8.groups(Order.ProductID.Category; avg(Evaluation.Score):Score)	/Group by category and calculate the average evaluation score of each category

A9	Category	Score
	Electric appliance	3.98
	Fruits	3.86

✦ 8.22 Cross Apply operation



Traverse multiple data files, perform Cross Apply operation between a table sequence and values of a sequence to generate a new table sequence.

Traverse all the online learning terminal questionnaires of a primary school stored in the folders, and calculate the proportion of each terminal.

▼	Primary School
>	Grade1
>	Grade2
▼	Grade3
	Class1
	Class2
	Class3
	Class4
	Class5
	Class6
>	Grade4
>	Grade5
>	Grade6

ID	STUDENT_NAME	TERMINAL
1	Rebecca Moore	Phone
2	Ashley Wilson	Phone,PC,Pad
3	Rachel Johnson	Phone,PC,Pad
4	Emily Smith	Phone,Pad
5	Ashley Smith	Phone,PC
6	Matthew Johnson	Phone
7	Alexis Smith	Phone,PC
8	Megan Wilson	Phone,PC,Pad
...

✦ 8.22 Cross Apply operation



Use A.news() function to perform Cross Apply operation between the table sequence and the split sequence of terminals. SPL script is as follows:

	A	B	C
1	=directory@ps("D:/Primary School")		/Recursively traverse all the files in the specified directory
2	for A1	=file(A2).xlsimport@t()	/Import the Excel file of each class 's questionnaire in loop
3		=@ += B2.len()	/Calculate the total number of rows, i.e. the total number of students
4		=B2.news(B2.TERMINAL.split@c(); ID, STUDENT_NAME, ~:TERMINAL)	/Use news function to perform Cross Apply operation between the questionnaire and the split sequence of terminals
5		=B4.groups(TERMINAL; count(~):Count) @	/Group by terminal and count the number of each terminal, merge the results to the current cell value. If you merge B4 directly, you may run out of memory.
6	=B5.groups(TERMINAL;string(sum(Count)/B3, "#.##%"):PERCENTAGE)		/Calculate the percentage of each type of terminal

A6

TERMINAL	PERCENTAGE
PC	70%
Pad	56.67%
Phone	93.33%

✦ 8.23 Outer Apply operation



Perform Outer Apply operation on a table sequence and values of a sequence to generate a new table sequence.

According to the *PostRecord* table, find the most commonly used labels of each author.

ID	TITLE	Author	Label
1	Easy analysis of Excel	Ashley	Excel,ETL,Import,Export
2	Early commute: Easy to pivot excel	Rachel	Excel,Pivot,Python
3	Initial experience of SPL	Rebecca	
4	Talking about set and reference	Emily	Set,Reference,Dispersed,SQL
5	Early commute: Better weapon than Python	Emily	Python,Contrast,Install
...

✦ 8.23 Outer Apply operation



SPL script is as follows, where @1 option is used with A.news() function to perform Outer Apply operation:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from PostRecord")	/Query PostRecord table
3	=A2.news@1(A2.Label.split@c(); ID,Title,Author,~:Label)	/Use @1 option with A.news function to perform Outer Apply operation on the post records and the split sequence of labels. Keep the post records even if the label does not exist
4	=A3.groups(Author,Label;count(~):Count)	/Group and count the number of each label for each author
5	=A4.group(Author).conj(~.maxp@a(Count))	/Group by author and select the most frequently used labels

A5	Author	Label	Count
	Rebecca	(null)	1
	Ashley	Excel	3
	Ashley	SPL	3
	Rachel	Python	4

✦ 8.24 Convert Apply operation to Cartesian product



Join fields of sequences to calculate cross product and generate a new table sequence.

Based on the *Teachers* table and *Courses* table, find the name of each teacher that potentially be able to teach a course.

Teachers		
Teacher	Branch	Courses
Petitti	Matematica	28,33,30,35
Canales	Apesca	11,16,12,17,13,18,14,19
Lucero	NavegacionI	6,11,16,21,7,12,17,22,...
Bergamaschi	TecPesc	1,26,2,27,3,28,4,29,5,30
...

Courses	
ID	Name
1	lua
2	maa
3	mia
4	jua
...	...

✦ 8.24 Convert Apply operation to Cartesian product



SPL script is as follows, where A.news() function is used to calculate the cross product of a table sequence with another sequence:

	A	B
1	=file("Teachers.txt").import@t().run(Courses=Courses.split@cp())	/Import <i>Teachers.txt</i> , and split Courses field to a sequence by comma
2	=file("Courses.txt").import@t()	/Import <i>Courses.txt</i>
3	=A2.news(A1;ID,Name:Course,Teacher,Courses)	/Use A.news function to calculate the cross product of <i>Courses.txt</i> and <i>Teachers.txt</i>
4	=A3.select(Courses.contain(ID))	/Select records where the course ID is included in the sequence of courses
5	=A4.group(Course;~.(Teacher).concat@c():Teachers)	/Group by course, and concatenate the sequence of teachers with commas to form the Teachers field

A5	Course	Teachers
	jua	Bergamaschi,Puebla,Jimenez
	jub	Lucero,Mazza,Puebla,Chiatti,Jimenez,Luceroo
	juc	Canales,Lucero,Mazza,Puebla,Chiatti,Luceroo

✦ 8.25 Complex uses of Apply operation



Perform join over three tables to generate a new table sequence and then group & aggregate.
Based on the associated *Employee* table, *Order* table and *Detail* table, give the salespeople whose actual amount of a single order exceeds 1000 a performance reward of 5% of the order amount.



✦ 8.25 Complex uses of Apply operation



A.news() function is used to perform join and calculation. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Order where year(Date)=2014")	/Query data of 2014 in <i>Order</i> table
3	=A1.query("select * from Detail")	/Query <i>Detail</i> table
4	=A1.query("select * from Employee")	/Query <i>Employee</i> table
5	=A2.switch(EmployeeID,A4:ID)	/Use switch function to switch EmployeeID in <i>Order</i> table to corresponding records in <i>Employee</i> table
6	=A3.group(ID)	/Group <i>Detail</i> table by order ID
7	=A6.news(A2.select(ID:A6.~.ID); EmployeeID,(s=sum(Amount*(1-Discount)), if(s>1000, s*1.05, s)):Amount)	/Use news function to join <i>Detail</i> table and <i>Order</i> table by order ID, and calculate the actual amount of each order
8	=A7.groups(EmployeeID.Name:Name; sum(Amount):Amount)	/Group records and calculate the total sales amount of each employee

A8	Name	Amount
	Alexis	358882.02
	Emily	432435.85



Chapter 9

SPL COOKBOOK

Inter-set operations

✦ 9.1 Concatenation of two sets



Concatenate and calculate the data in two tables of the same structure.

According to the sales records in 2014 and 2015, calculate the total number of orders per customer in the two years.

ID	Customer	Date	Amount
10400	EASTC	2014/01/01	3063.0
10401	RATTC	2014/01/01	3868.6
10402	ERNSH	2014/01/02	2713.5
...

✦ 9.1 Concatenation of two sets



SPL script is as follows, where symbol "|" is used to calculate the concatenation:

	A	B
1	=file("S2014.txt").import@t(Customer)	/Import customers of 2014
2	=file("S2015.txt").import@t(Customer)	/Import customers of 2015
3	=A1 A2	/Use " " to concatenate customers, including the duplicate ones, of the two years.
4	=A3.groups(Customer; count(~):Count)	/Count the orders of each customer

A4	Customer	Count
	ANATR	5
	ANTON	6

✦ 9.2 Intersection of two sets



Calculate the intersection of two sets.

In the after-school classes registration table, query which students sign up for both painting class and dance class.

ID	StudentID	Subject
1	2	Painting
2	4	Dance
3	3	Robot
4	2	Dance
5	5	Writing
...

✦ 9.2 Intersection of two sets



SPL script is as follows, where symbol "^" is used to calculate the intersection.

	A	B
1	=file("Interest.txt").import@t()	/Import the text file
2	=A1.select(Subject:"Painting")	/Get records of painting
3	=A1.select(Subject:"Dance")	/Get records of dancing
4	=A2.(StudentID) ^ A3.(StudentID)	/Use "^" to get intersection of students who sign up for painting and dancing

A4	Member
	2
	8
	11
	...

✦ 9.3 Union of two sets



Calculate the union of two sets.

Based on the after-school class registration table, query which students sign up for the painting class or dance class.

ID	StudentID	Subject
1	2	Painting
2	4	Dance
3	3	Robot
4	2	Dance
5	5	Writing
...

✦ 9.3 Union of two sets



SPL script is as follows, where symbol "&" is used to calculate the union:

	A	B
1	=file("Interest.txt").import@t()	/Import the text file
2	=A1.select(Subject:"Painting")	/Get records of painting
3	=A1.select(Subject:"Dance")	/Get records of dancing
4	=A2.(StudentID) & A3.(StudentID)	/Use "&" to get the union of students who sign up for painting or dancing

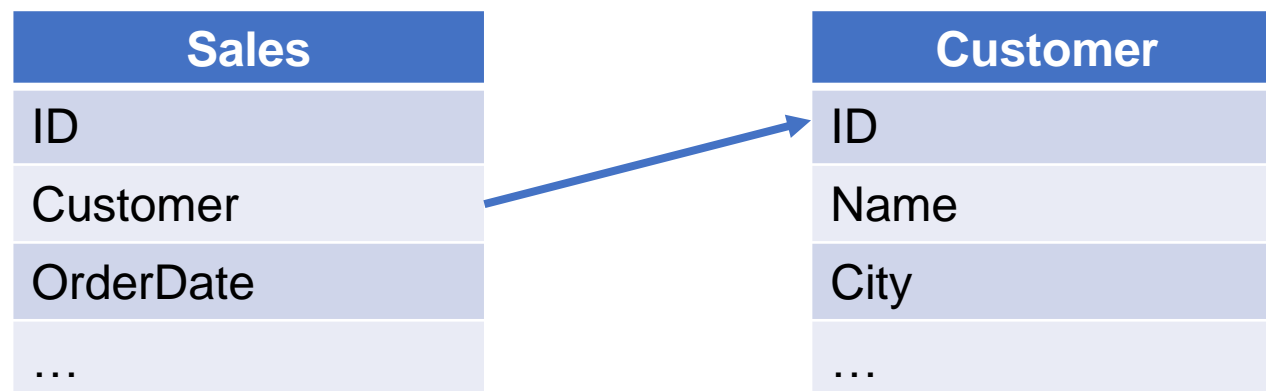
A4	Member
	2
	4
	8
	...

✦ 9.4 Difference of two sets



Calculate the difference of two sets.

Query the new customers according to the sales table and customer table.



✦ 9.4 Difference of two sets



SPL script is as follows, where symbol "\" is used to calculate the difference:

	A	B
1	=connect("db")	/Connect to the database
2	=A1.query("select * from Sales where year(OrderDate)=2014")	/Get sales records of 2014
3	=A1.query("select * from Customer")	/Get records from <i>Customer</i> table
4	=A2.id(Customer)	/Use id function to remove duplicate sales records to get a sequence of unique customers
5	=A3.(ID)	/Get the sequence of customer IDs from <i>Customer</i> table
6	=A4\A5	/Use "\" to get the difference

A6	Member
	DOS
	HUN
	URL

Note: This example is for explaining how to perform a difference operation. Actually it's more convenient to get same result using A.switch@d()/A.join@d(), which perform a join and filtering.

✦ 9.5 XOR operation of two sets



The XOR operation (symmetric difference) of two sets.

According to the different score records of the two semesters, find the student IDs whose total scores rank in top 10 only once in both the first and second semesters.

CLASS	STUDENTID	SUBJECT	SCORE
Class one	1	English	84
Class one	1	Math	77
Class one	1	PE	69
Class one	2	English	81
Class one	2	Math	80
...

✦ 9.5 XOR operation of two sets



SPL script is as follows, where the XOR operator "%" is used.

	A	B
1	=file("Scores1.csv").import@ct()	/Import scores of the first semester
2	=file("Scores2.csv").import@ct()	/Import scores of the second semester
3	=A1.groups(STUDENTID; sum(SCORE):Score)	/Group by students and sum their total scores in the first semester
4	=A2.groups(STUDENTID; sum(SCORE):Score)	/Group by students and sum their total scores in the second semester
5	=A3.top(-10;Score).(STUDENTID)	/Get student IDs whose total scores rank in top 10 in the first semester
6	=A4.top(-10;Score).(STUDENTID)	/Get student IDs whose total scores rank in top 10 in the second semester
7	=A5%A6	/Get unique student IDs from A5 and A6

A5	Member
	2
	9
	4
	10
	...

A6	Member
	12
	1
	8
	4
	...

A7	Member
	2
	9
	10
	7
	...

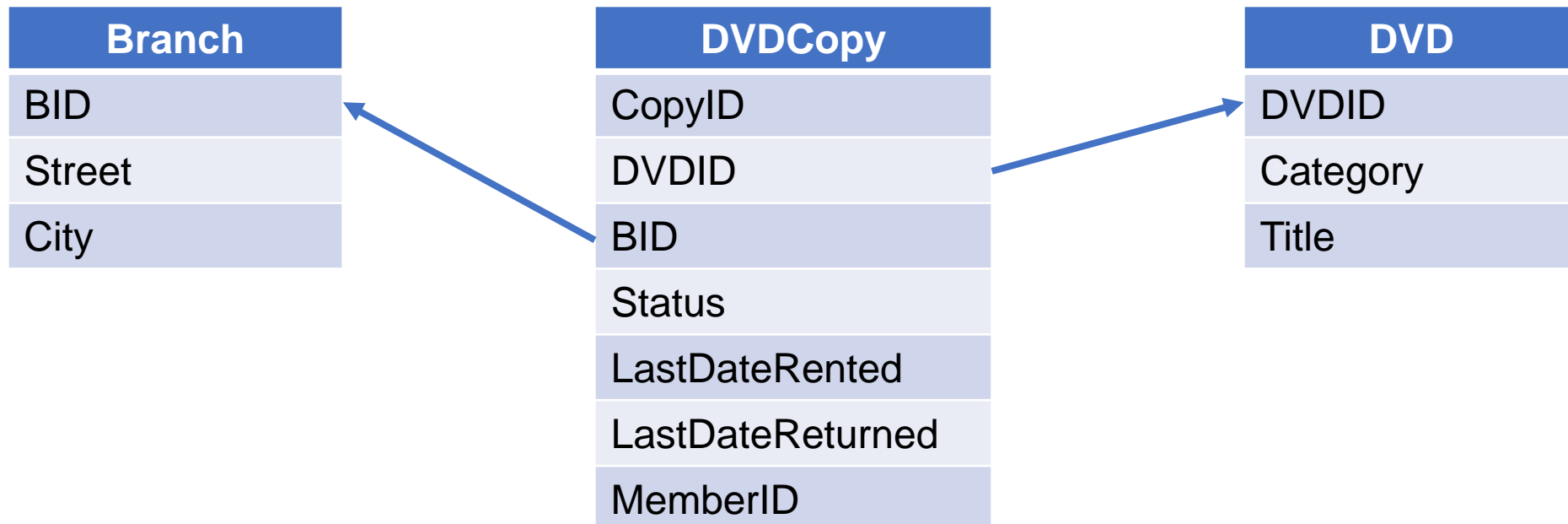
✦ 9.6 Mixed use of concatenation and difference



Calculate the concatenation and difference of multiple sets.

Query the branch stores with less than 4 categories of DVD copies:

Branch table stores the information of DVD branch stores; *DVD* table stores the title and category information of DVD; *DVDCopy* table stores multiple copies of DVD. DVD copies are the information of concrete discs stored in each branch store.



✦ 9.6 Mixed use of concatenation and difference



SPL script is as follows, in which the concatenation operator | and the difference operator \ are used:

	A	B
1	=connect("db")	/Connect to database
2	=Branch=A1.query("select * from Branch")	/Read the branch information and define it as Branch variable
3	=DVD=A1.query("select * from DVD")	/Read DVD information and define it as DVD variable
4	=DVDCopy=A1.query("select * from DVDCopy")	/Read the DVDCopy information and define it as DVDCopy variable
5	=DVDCopy.switch(DVDID,DVD:DVDID; BID,Branch:BID)	/Convert the DVDID field of DVDCopy into the corresponding records in DVD, and the BID field to the corresponding records in Branch
6	=DVDCopy.select(STATUS!="Miss" && LASTDATERETURNED!=null)	/Filter away lost and unreturned DVD copies
7	=A6.group(BID)	/Group the filtered data by BID
8	=A7.select(~.icount(DVDID.CATEGORY)<4)	/Select branches with DVD copies less than 4 categories
9	=A8.(BID) (Branch \ A7.(BID))	/Branches with out of stock DVD copies. A8.(BID) refers to the branches whose categories of DVD copies are less than 4, and Branch \ A7.(BID) the branches that never have any DVD copies

A9	BID	STREET	CITY
	B002	Street2	Houston
	B003	Street3	LA
	B004	Street4	Lincoln

✦ 9.7 Set operations of sequences: intersection and union



Calculate the intersection and union of sequences.

List municipalities directly under the central government and the first-tier cities, and get cities that are both municipalities and the first-tier cities. In China, municipalities are Beijing, Tianjin, Shanghai and Chongqing; the first-tier cities include Shanghai, Beijing, Shenzhen and Guangzhou.

✦ 9.7 Set operations of sequences: intersection and union



SPL script is as follows:

	A	B
1	[Beijing,Tianjin,Shanghai,Chongqing]	/Municipalities directly under the Central Government
2	[Shanghai,Beijing,Shenzhen,Guangzhou]	/The first-tier cities
3	=A1&A2	/Get the union of municipalities and the first-tier cities
4	=A1^A2	/Get the intersection, i.e. the-first tier cities among the municipalities

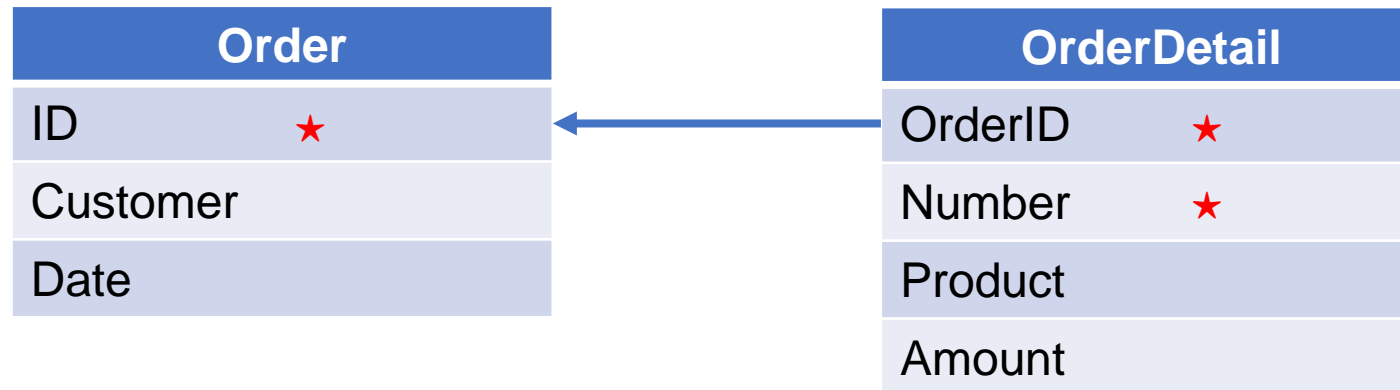
A3	Member
	Beijing
	Tianjin
	Shanghai
	Chongqing
	Shenzhen
	Guangzhou

A4	Member
	Beijing
	Shanghai

✦ 9.8 Concatenation of all set members in a sequence



Calculate the concatenation in an aggregate operation over a sequence.
The relationship of *Order* table and *OrderDetail* table are that of main table and sub table. Each Order record corresponds to multiple OrderDetail records.



The OrderDetail records vary in length. **Task:** to get the following table:

ID	Customer	Date	Product1	Amount1	Product2	Amount2	Product3	Amount3
1	3	20190101	Apple	5	Milk	3	Salt	1
2	5	20190102	Beef	2	Pork	4		
3	2	20190102	Pizza	3				

✦ 9.8 Concatenation of all set members in a sequence



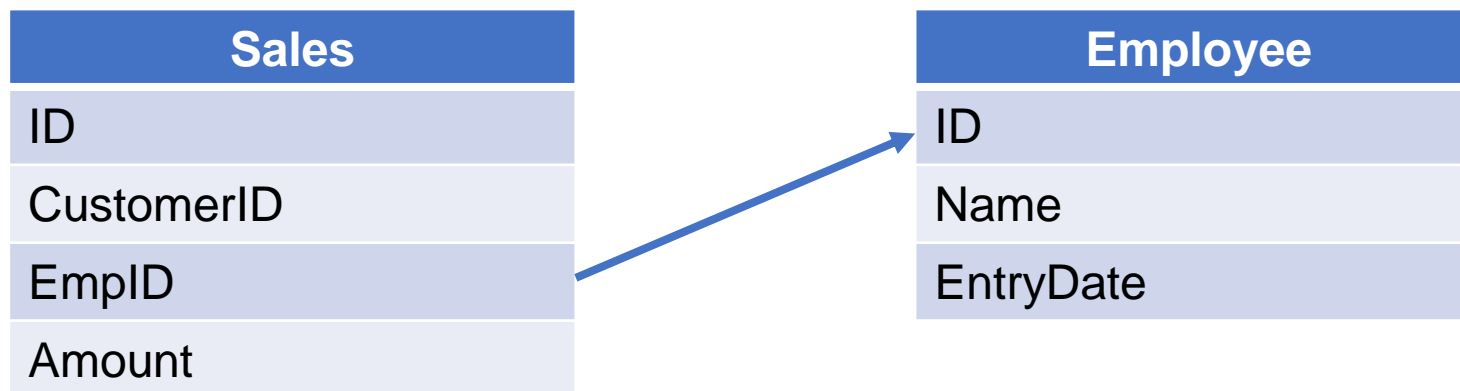
SPL script is as follows, where A.conj() function is used to concatenate sequence-type members:

	A	B
1	=connect("db")	/Connect to the database
2	=A1.query("select * from OrderDetail left join Order on Order.ID=OrderDetail.OrderID")	/Import the two tables and left join <i>Order</i> table by order IDs
3	=A2.group(ID)	/Group retrieved records by order ID
4	=A3.max(~.count()).("Product"+string(~)+","+"Amount"+string(~)).concat@c()	/Calculate the maximum number of members, and then generate data structure strings
5	=create(ID,Customer,Date,\$A4)	/Create a table sequence according to the data structure determined by A4
6	>A3.run(A5.record([ID,Customer,Date] ~.([Product,Amount]).conj()))	/Loop through the groups to piece members together into a sequence and concatenate Product and Amount from these groups using conj() function, and then insert the complete records into A5's table sequence

✦ 9.9 The union of all set members in a sequence



Select two sets of records from a table according to different conditions, and calculate their union. Based on the following two tables, we want to find employees who have worked in the company for less than one year and who are among the bottom 10% in terms of performances.



✦ 9.9 The union of all set members in a sequence



SPL script is as follows, where A.union() function gets the union of the records in all table sequences and returns a record sequence:

	A	B
1	=connect("db")	/Connect to data source
2	=A1.query("select * from Employee")	/Read <i>Employee</i> table
3	=A1.query("select * from Sales")	/Read <i>Sales</i> table
4	=A2.select(age(EntryDate)<1)	/Find employees whose employment duration is less than one year
5	=A3.groups(EmpID; sum(Amount):Amount)	/Group <i>Sales</i> table by EmpID and calculate the total sales amount for each
6	=A5.top(A5.len()/10; Amount)	/Get records where the employees' sales amounts are among the bottom 10%
7	=A2.join@i(ID,A6:EmpID)	/Perform a filtering join on <i>Employee</i> table and A6's records
8	=A4.union()	/Get union of A4's set and A7's set, which are all the eligible employees

A8	ID	Name	EntryDate
	89	Emily	2020/02/01
	241	Samantha	2020/01/01

✦ 9.10 Merge same-order sets in the current order to calculate concatenation



For two tables of same structure, merge their records in order according to multiple fields. Scores of math and English are stored in two files. Calculate the total score of each student.

Math:

CLASS	STUDENTID	SUBJECT	SCORE
1	1	Math	77
1	2	Math	80
...

English:

CLASS	STUDENTID	SUBJECT	SCORE
1	1	English	84
1	2	English	81
...

✦ 9.10 Merge same-order sets in the current order to calculate concatenation



The SPL script uses A.merge(xi, ...) function to concatenate table sequences by expressions xi, ... :

	A	B
1	=file("Math.txt").import@t()	/Import <i>Math</i> .txt
2	=file("English.txt").import@t()	/Import <i>English</i> .txt
3	=A1.sort(CLASS,STUDENTID)	/Sort <i>Math</i> table by CLASS and STUDENTID
4	=A2.sort(CLASS,STUDENTID)	/Sort <i>English</i> table by CLASS and STUDENTID
5	=A3.A4.merge(CLASS,STUDENTID)	/merge() function is used to merge records in order by CLASS and STUDENTID
6	=A5.groups@o(CLASS,STUDENTID; ~.sum(SCORE):TOTALSCORE)	/Use groups@o() to group records, which creates a new group whenever the next value changes, and sums scores for each student

A6	CLASS	STUDENTID	TOTALSCORE
	1	1	161
	1	2	161
	1	3	159

✦ 9.11 Merge same-order sets to calculate union



For two tables of same structure, merge their records in order according to values of a specified field and remove the duplicate records.

Sales records are stored in *Online* table and *Store* table according to distribution channels. They are of same structure. Records during promotion periods of both channels are stored in both tables. Calculate the actual total sales.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 9.11 Merge same-order sets to calculate union



The SPL script uses @u option of A.merge(xi, ...) function to remove duplicate records during the order-based merge:

	A	B
1	=file("Online.txt").import@t()	/Import <i>Online.txt</i>
2	=file("Store.txt").import@t()	/Import <i>Store.txt</i>
3	=A1.sort(OrderID)	/Sort <i>Online</i> table by OrderID
4	=A2.sort(OrderID)	/Sort <i>Store</i> table by OrderID
5	=A3.A4.merge@u(OrderID)	/Use @u option with the merge function to merge two tables by OrderID and delete duplicates at the same time
6	=A5.sum(Amount)	/Sum the sales amounts

A6	Value
	678756.41

✦ 9.12 Merge same-order sets to calculate intersection



For two tables of same structure, merge their records in order according to values of the specified field, and only keep the duplicate ones.

According to online sales records and store sales records, find out how many online and offline sales records are repeatedly saved.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 9.12 Merge same-order sets to calculate intersection



SPL script is as follows, where @i option of A.merge(xi, ...) function is used to return a table sequence consisting of common members of A(i):

	A	B
1	=file("Online.txt").import@t()	/Import <i>Online.txt</i>
2	=file("Store.txt").import@t()	/Import <i>Store.txt</i>
3	=A1.sort(OrderID)	/Sort <i>Online</i> table by OrderID
4	=A2.sort(OrderID)	/Sort <i>Store</i> table by OrderID
5	=A3.merge@i(A4, OrderID)	/Use @i option with the merge function to merge the two tables in order by order ID, and return the common members
6	=A5.count()	/Count the common records

A6	Value
	70

✦ 9.13 Merge same-order sets to calculate XOR



For two tables of same structure, merge their records in order according to values of the specified field, and only keep the non-duplicate ones.

Compare two random sampling files and list different sequence numbers.

ID	Predicted_Y	Original_Y
10	0.012388464367608093	0.0
11	0.01519899123978988	0.0
13	0.0007920238885061248	0.0
19	0.0012656367468159102	0.0
21	0.009460545997473379	0.0
23	0.024176791871681664	0.0
...

✦ 9.13 Merge same-order sets to calculate XOR



SPL script is as follows, where @x option of A.merge(xi, ...) function is used to return a table sequence consisting of members of A(i) after removing the common members:

	A	B
1	=file("p1.txt").import@t()	/Read the first sampling file
2	=file("p2.txt").import@t()	/Read the second sampling file
3	=A1.sort(ID)	/Sort the first file by ID
4	=A2.sort(ID)	/Sort the second file by ID
5	=A3.merge@x(ID)	/Use @x option with the merge function to merge the files in order by ID and return records with different IDs.
6	=A5.len()	/Return the number of different IDs

A6	Value
	458

✦ 9.14 Merge same-order sets to calculate difference



For two tables of same structure, merge their records in order according to values of the specified field and query differences.

Two transaction information files based on different versions are stored as *new.csv* and *old.csv*. Find out the newly-added, deleted and modified records respectively.

old.csv

UserName	Date	SaleValue	SaleCount
Rachel	2015-03-01	4500	9
Rachel	2015-03-03	8700	4
Tom	2015-03-02	3000	8
Tom	2015-03-03	5000	7
Tom	2015-03-04	6000	12
John	2015-03-02	4000	3
John	2015-03-02	4300	9
John	2015-03-04	4800	4

new.csv

UserName	Date	SaleValue	SaleCount
Rachel	2015-03-01	4500	9
Rachel	2015-03-02	5000	5
Ashley	2015-03-01	6000	5
Rachel	2015-03-03	11700	4
Tom	2015-03-03	5000	7
Tom	2015-03-04	6000	12
John	2015-03-02	4000	3
John	2015-03-02	4300	9
John	2015-03-04	4800	4

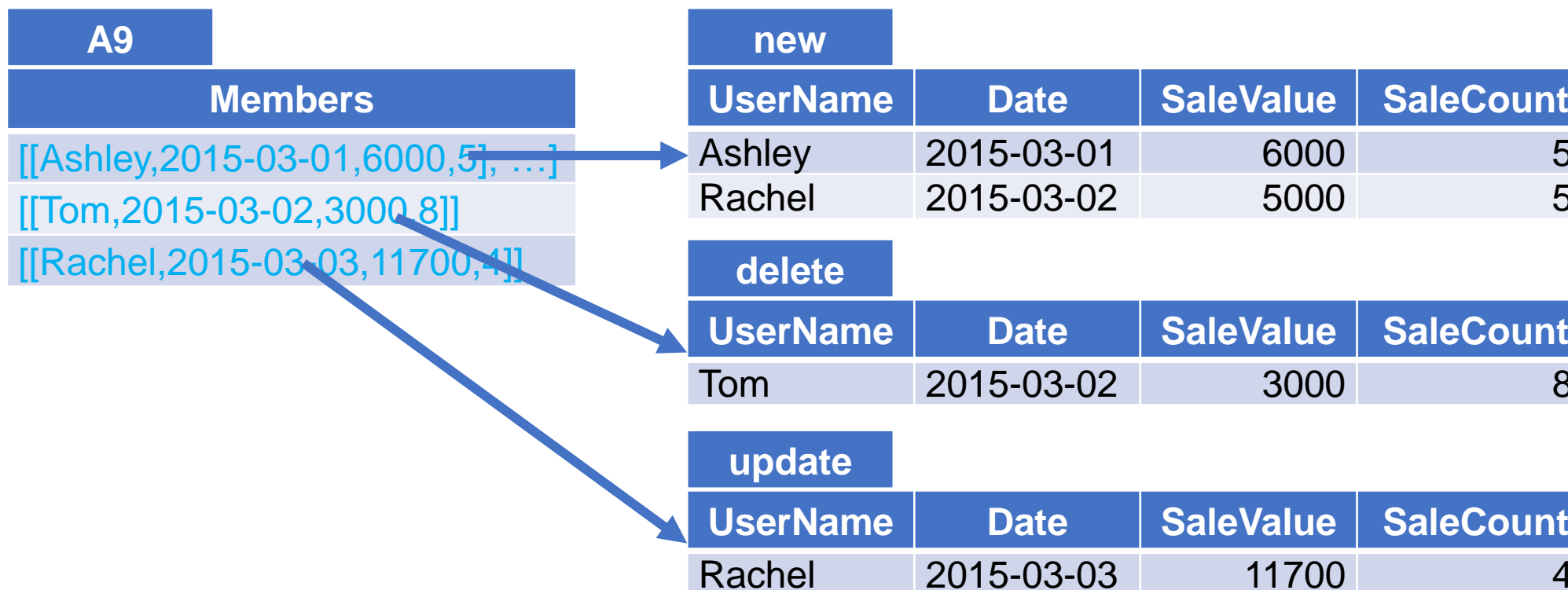
✦ 9.14 Merge same-order sets to calculate difference



The SPL script uses @d option of A.merge(xi, ...) function to remove members of A(2) &...A(n) from A(1) to generate a new table sequence:

	A	B
1	=file("old.csv").import@ct()	/Import <i>old.csv</i>
2	=file("new.csv").import@ct()	/Import <i>new.csv</i>
3	=A1.sort(UserName,Date)	/Sort <i>old</i> table by UserName and Date
4	=A2.sort(UserName,Date)	/Sort <i>new</i> table by UserName and Date
5	=new=[A4,A3].merge@d(UserName,Date)	/merge@d() deletes records of A3 from A4 while performing order-based merge to generate a sequence of new records
6	=delete=[A3,A4].merge@d(UserName,Date)	/merge@d() deletes records of A4 from A3 while performing order-based merge to generate a sequence of deleted records
7	=diff=[A4,A3].merge@d(UserName,Date,SaleValue,SaleCount)	/merge@d() gets records of A4 where the specified field values differ from A3 while performing order-based merge
8	=update=[diff,new].merge@d(UserName,Date)	/merge@d() deletes new records from different records while performing order-based merge to generate a sequence of updated records
9	return [new, delete, update]	/Return a sequence consisting of sequences of new, deleted and updated records

✦ 9.14 Merge same-order sets in the current order to calculate difference



✦ 9.15 Merge table sequences by primary key to calculate concatenation



For tables of same structure, merge their records in order according to the primary key.

According to the body temperature records from June 1 to 20, get a list of students who have had a fever for at least 3 days consecutively.

StudentID	Name	Fever
10	Ryan	0
5	Ashley	0
13	Daniel	1
19	Samantha	0
1	Rebecca	0
...

✦ 9.15 Merge table sequences by primary key to calculate concatenation



The SPL script uses A.merge() function to perform an order-based merge by the primary key as long as the primary key is set for A(i):

	A	B
1	=to(601, 620)	/Create a sequence of file names
2	=A1.(file(string(~)+".txt").import@t())	/Import files from June 1 to June 20 in loop
3	=A2.(~.keys(StudentID).sort(StudentID))	/Set StudentID as the primary key and sort the files by the key
4	=A3.merge()	/merge() compares the primary key values to perform the order-based merge
5	=A4.group@o(StudentID,Fever)	/group@o() creates a new group whenever the key value changes
6	=A5.select(~.Fever==1 && ~.len()>=3).id(Name)	/Get students who have had a fever for at least 3 days consecutively

A6	Name
	Ashley
	Rachel

✦ 9.16 Merge table sequences to find differences



Find how many rows of data are different between two data files of same structure.
Find how many rows of data are different between two CSV files.

ID	Predicted_Y	Original_Y
10	0.012388464367608093	0.0
11	0.01519899123978988	0.0
13	0.0007920238885061248	0.0
19	0.0012656367468159102	0.0
21	0.009460545997473379	0.0
23	0.024176791871681664	0.0
...

✦ 9.16 Merge table sequences to find differences



SPL script uses `A.merge()` function to compare all fields to perform the order-based merge when no primary key is set for `A(i)`:

	A	B
1	<code>=file("p1.txt").import@t()</code>	<code>/Import the first sampling file <i>p1</i></code>
2	<code>=file("p2.txt").import@t()</code>	<code>/Import the second sampling file <i>p2</i></code>
3	<code>=A1,A2].merge@x()</code>	<code>/merge() compares all fields to perform the order-based merge. @x option returns a sequence of different IDs, that is, the records with different IDs</code>
4	<code>=A3.len()</code>	<code>/Return the number of different records</code>

A4	Value
	458

✦ 9.17 Merge unordered tables to calculate union



Merge records in two tables of same structure and calculate sum. The tables have common records and are unordered.

According to two sales record tables db1 and db2 of same structure in the database, calculate the total sales amount in 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10426	GALED	4	2014/01/27	338.2
10676	TORTU	2	2014/09/22	534.85
10390	ERNSH	6	2013/12/23	2275.2
10400	EASTC	1	2014/01/01	3063.0
10464	FURIB	4	2014/03/04	1848.0
...

✦ 9.17 Merge unordered tables to calculate union



SPL script is as follows, where @o option is used with A.merge(xi, ...) function. It does not assume that A(i) is ordered by [xi,...] :

	A	B
1	=connect("db1").query("select * from Sales")	/Query Sales table from db1
2	=connect("db2").query("select * from Sales")	/Query Sales table from db2
3	=A1,A2].merge@ou(OrderID)	/merge() performs the order-based merge by OrderID. @o option indicates that the records are not necessarily ordered by OrderID; @u option removes records with duplicate IDs
4	=A3.select(year(OrderDate)==2014)	/Get records of 2014
5	=A4.sum(Amount)	/Calculate the total sales in 2014

A5	Value
	723388.75

✦ 9.18 Aggregation of sequences: union & difference



Calculate union or difference of sequence-type members in a sequence.

According to the course table and course selection table, query which courses are not selected in the course table.

Course		
ID	NAME	TEACHERID
1	Environmental protection and ...	5
2	Mental health of College Students	1
3	Computer language Matlab	8
4	Electromechanical basic practice	7
5	Introduction to modern life science	3
6	Modern wireless communication system	14
...

SelectCourse		
ID	STUDENTID	COURSE
1	59	2,7
2	43	1,8
3	52	2,7,10
4	44	1,10
5	37	5,6
6	57	3
...

✦ 9.18 Aggregation of sequences: union & difference



A.union() function calculates the union of all sequence-type members in a sequence;

A.diff() function calculates the difference of all sequence-type members in a sequence.

SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Course")	/Query the <i>Course</i> table
3	=A1.query("select * from SelectCourse")	/Query the <i>SelectCourse</i> table
4	=A3.union(COURSE.split@cp())	/Split selected courses in <i>SelectCourse</i> table by comma and get union of theses course sequences using union() function
5	=A2.(ID)	/Get course IDs from the <i>Course</i> table
6	=A2(A5.pos([A5,A4].diff()))	/Get difference of course IDs in the two tables, i.e. the courses that no students select; then find their positions in A5 and get them from A2

A6		
ID	NAME	TEACHERID
1	Fundamentals of economic management	21

✦ 9.19 Aggregation of sequences:intersection



Calculate intersection of sequence-type members in a sequence.

According to the sales table, find the customers whose order amounts rank in top 20 in each month of 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 9.19 Aggregation of sequences:intersection



SPL script is as follows, where A.isect() function is used to calculate the intersection of all sequence-type members:

	A	B
1	=connect("db").query("select * from sales")	/Connect to data source to query the <i>sales</i> table
2	=A1.select(year(OrderDate)=2014)	/Select records of 2014
3	=A2.group(month(OrderDate))	/Use group() function to group records of 2014 by month
4	=A3.(~.group(Customer))	/Group each group again by customer
5	=A4.(~.top(-20;sum(Amount)))	/Loop through records of each month to find customers whose order amounts rank in top 20 in each month
6	=A5.(~.(Customer))	/List the top 20 customers in each month
7	=A6.isect()	/Get intersection of groups using isect() function

A7	Member
	HANAR
	SAVEA

✦ 9.20 Perform mixed set operations over two small files



Perform the required set operations based on two text files.

According to the members tables of running club and ball club, get all members in the two clubs, members who participate in at least one club, members who participate in two clubs at the same time, and members who only participate in running club.

✦ 9.20 Perform mixed set operations over two small files



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/running.txt").import@t().([NAME,SURNAME])</code>	/Members of running club
2	<code>=file("E:/txt/ball.txt").import@t().([NAME,SURNAME])</code>	/Members of ball club
3	<code>=A1 A2</code>	/Concatenation, all members of the two clubs
4	<code>=A1&A2</code>	/Union, members who join at least one club
5	<code>=A1^A2</code>	/Intersection, members who join both clubs
6	<code>=A1\A2</code>	/Difference, members who only join the running club

A3

member
[Joshua,Johnson]
[Sophia,Williams]
...

A4

member
[Joshua,Johnson]
[Sophia,Williams]
...

A5

member
[Joshua,Johnson]
[Zachary,Williams]
...

A6

member
[Sophia,Williams]
[Christopher,Johnson]
...

✦ 9.21 Perform complex set operations over two small files



Perform complex set operations over text files.

The user login information is stored in different files by month. Query users who log in at least once in three months, users who log in every month within three months, and users who log in only in January.

✦ 9.21 Perform complex set operations over two small files



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/user_login_info_1.txt").import@t().group@1(userid)</code>	/Users' first login information in January
2	<code>=file("E:/txt/user_login_info_2.txt").import@t().group@1(userid)</code>	/Users first login information in Feb
3	<code>=file("E:/txt/user_login_info_3.txt").import@t().group@1(userid)</code>	/Users first login information in Mar
4	<code>=[A1,A2,A3].merge@u(userid)</code>	/Union, users who log in at least once in 3 months
5	<code>=[A1,A2,A3].merge@i(userid)</code>	/Intersection, users who log in every month for 3 months
6	<code>=[A1,A2,A3].merge@d(userid)</code>	/Difference, users who log in only in January

A4	
userid	login
600001	2019-01-25 02:49:43
600002	2019-01-20 03:00:28
600003	2019-01-13 14:34:20
...	...

A5	
userid	login
600001	2019-01-25 02:49:43
600002	2019-01-20 03:00:28
600003	2019-01-13 14:34:20
...	...

A6	
userid	login
601293	2019-01-10 08:04:36
601999	2019-01-11 16:49:25
605227	2019-01-18 15:24:26
...	...

✦ 9.22 Merge two big data tables to calculate concatenation



Merge and calculate two big data tables of same structure.

There are sales record tables of same structure in db1 and db2 respectively. The data volume is too large to be loaded into memory. Calculate the monthly sales records count in 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 9.22 Merge two big data tables to calculate concatenation



The SPL script uses CS.mergex(xi, ...) function to merge sequences of records in cursors:

	A	B
1	=connect("db1").cursor("select * from Sales where year(OrderDate)=2014 order by OrderDate")	/Query <i>Sales</i> table of 2014 in db1 and sort it by OrderDate
2	=connect("db2").cursor("select * from Sales where year(OrderDate)=2014 order by OrderDate")	/Query <i>Sales</i> table of 2014 in db2 and sort it by OrderDate
3	=A1,A2].mergex(OrderDate)	/mergex() merges the two cursors by OrderDate
4	=A3.groups@o(month(OrderDate):Month; count(~):Count)	/groups() groups and summarize sales records count for each month. @o option creates a new group whenever the month changes

A5	Month	Count
	1	33
	2	29

✦ 9.23 Merge two big data tables to calculate union



Merge and calculate two big data tables of same structure. The tables have some common records.

There are sales record tables of same structure in db1 and db2 respectively. The data volume is too large to be loaded into memory. Calculate the total order amount of each customer in 2014.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 9.23 Merge two big data tables to calculate union



CS.mergex(xi, ...) function supports a number of options, such as @u, @i, @d and @x, which work similarly to options for A.merge().

	A	B
1	=connect("db1").cursor("select * from Sales where year(OrderDate)=2014 order by OrderID")	/Query <i>Sales</i> table of 2014 in db1 and sort it by OrderID
2	=connect("db2").cursor("select * from Sales where year(OrderDate)=2014 order by OrderID")	/Query <i>Sales</i> table of 2014 in db2 and sort it by OrderID
3	=[A1,A2].mergex@u(OrderID)	/mergex@u() removes duplicate records while merging the cursors by OrderID
4	=A3.groups(Customer; sum(Amount):Amount)	/Use groups() function to group and summarize each customer's sales amount

A5	Customer	Amount
	ANATR	1129.75
	ANTON	6452.15



Chapter 10

SPL COOKBOOK

Transposition

✦ 10.1 Row to column transposition



Row to column transposition: Merge multiple rows into one row, during which specific values of a specified column are converted into new columns. The new columns get their values from another column in the original rows.

Find the highest score of each subject in each class based on *score* table.

Class	StudentID	Subject	Score
Class one	1	Math	89
Class one	1	Chinese	93
Class two	2	Math	92
Class two	2	Chinese	97



Database pivot function supports row to column conversion

Class	MathMax	ChineseMax
Class one	89	93
Class two	92	97

✦ 10.1 Row to column transposition



The SQL query:

```
select * from ( select Class, Subject, Score from StudentScore )
pivot (
    max(Score) for Subject in (
        'Math' as MathMax, 'Chinese' as ChineseMax
    )
)
```

This is an example of Oracle. Not all databases have pivot functions. Pivot is only supported in new versions of mainstream databases.

✦ 10.1 Row to column transposition



Row to column transposition in SPL: first, get the highest score of each subject from the database.

Class	Subject	MaxScore
Class one	Math	89
Class one	Chinese	93
Class two	Math	92
Class two	Chinese	97



Then use SPL pivot function to perform the row to column conversion.

Class	MathMax	ChineseMax
Class one	89	93
Class two	92	97

✦ 10.1 Row to column transposition



The SPL script is as follows:

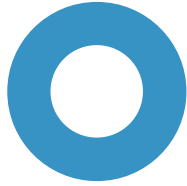
	A
1	=connect("oracle")
2	=A1.query("select Class, Subject, max(Score) MaxScore from StudentScore group by Class, Subject")
3	=A2.pivot(Class; Subject, MaxScore; "Math":"MathMax", "Chinese":"ChineseMax")

A1:Connect to database.

A2:The highest score of each subject in each class is directly retrieved from the database.

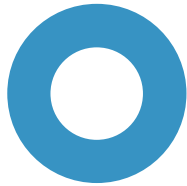
A3: Use pivot function to implement row to column conversion.

✦ 10.1 Row to column transposition



Not all databases support pivot.

MySQL, for example, does not have pivot function. It uses a subquery to perform group and aggregate and then a left join to achieve row to column conversion.



SPL's pivot is more adaptable.

It is difficult for SQL to handle an inter-column calculation after the transposition. And SQL cannot deal with multiple data sources scenarios.

✦ 10.2 Column to row transposition



Column to row transposition: Each value of column to be transposed generates a new row. The column name or corresponding name becomes the value of a new column, and the original column value becomes the value of another new column. Generate subject score table based on the summary score table.

StudentID	Math	Chinese
1	89	93
2	92	97



Database unpivot function supports column to row conversion.

StudentID	Subject	Score
1	Math	89
1	Chinese	93
2	Math	92
2	Chinese	97

✦ 10.2 Column to row transposition



The SQL query in Oracle:

```
select
    *
from
    StudentScore
unpivot
    (Score for Subject in (Math, Chinese))
```

✦ 10.2 Column to row transposition



The SPL `pivot@r()` function supports column to row transposition:

StudentID	Math	Chinese
1	89	93
2	92	97
3	91	88

`pivot@r()` →

StudentID	Subject	Score
1	Math	89
1	Chinese	93
2	Math	92
2	Chinese	97
3	Math	91
3	Chinese	88

✦ 10.2 Column to row transposition



The SPL script is as follows:

	A
1	=connect("oracle")
2	=A1.query("select * from StudentScore")
3	=A2.pivot@r(StudentID; Subject, Score; Math:"Math", Chinese:"Chinese")

A1:Connect to database.

A2:Read *StudentScore* table.

A3:Use the @r option with the pivot function to implement column to row transposition.

✦ 10.3 Dynamic row to column transposition



Dynamic row to column transposition: Names and number of transposed columns cannot be specified in advance. It need to be determined dynamically according to the values of the original column.

For example, here's the employee table:

Name	Dept	Area	Salary
David	Sales	Beijing	8000
Daniel	R&D	Beijing	15000
Andrew	Sales	Shanghai	9000
Robert	Sales	Beijing	26000
Rudy	R&D	Shanghai	23000
...

Calculate the average salary of each department in different areas. Now we don't know the names of areas but want to convert them in the following format:

Dept	Beijing	Shanghai	...
Sales	13000	11000	...
R&D	15000	14000	...

✦ 10.3 Dynamic row to column transposition



This is a row to column transposition. Name of the target transposed fields need to be extracted from the original data. The SPL pivot function will automatically extract the names of the target fields if their names are not specified.

SPL script is as follows:

	A
1	=connect("db")
2	=A1.query("select Dept,Area,avg(Salary) as AvgSalary from Employee group by Dept,Area")
3	=A2.pivot(Dept; Area, AvgSalary)

A1:Connect to data source.

A2:Get the average salaries grouped by department and area from the employee table.

A3: The pivot function performs row to column conversion, where the target fields are not specified.

✦ 10.4 Bidirectional transposition



Bidirectional transposition: Convert both rows to columns and columns to rows at the same time.

The sales table classified by channel and recorded by date is as follows:

Day	Online	Store
20190101	2400	1863
20190102	1814	670
20190103	3730	1444

Desired result:


Category	20190101	20190102	20190103
Online	2400	1814	3730
Store	1863	670	1444

✦ 10.4 Bidirectional transposition




First, perform column to row transposition to convert Online and Store to values under the new Category column.

Day	Online	Store
20190101	2400	1863
20190102	1814	670
20190103	3730	1444



Day	Category	Sales
20190101	Online	2400
20190101	Store	1863
20190102	Online	1814
20190102	Store	670
20190103	Online	3730
20190103	Store	1444

Then perform row to column transposition to convert the unique values of the Day column to the new column names.



Category	20190101	20190102	20190103
Online	2400	1814	3730
Store	1863	670	1444

✦ 10.4 Bidirectional transposition



The SPL script is as follows:

	A
1	=connect("db")
2	=A1.query("select * from Sales")
3	=A2.pivot@r(Day; Category, Sales)
4	=A3.pivot(Category; Day, Sales)

A3:Use pivot@r to perform column to row transposition to convert channel types (Online and Store) to values of Category column.

A4:Use pivot to perform row to column transposition to convert the unique Day values to column names.

✦ 10.4 Bidirectional transposition



Pivot function is suitable for static transposition

For static transposition (the table structure after transposition can be determined beforehand), you can use SPL pivot and pivot@r functions.

For the dynamic transposition for which the table structure after transposition cannot be decided in advance, SPL pivot function is sometimes incompetent. We will provide other solutions in later chapters.

✦ 10.5 Row to column transposition with dynamic columns by filling into a table



Convert rows into columns dynamically, generate column names dynamically based on record calculation, and finally fill all data into the new table sequence.

According to the income details, calculate all incomes of different sources for each employee. The categories are automatically generated:

Name	Source	Income
David	Salary	8000
David	Bonus	15000
Daniel	Salary	9000
Andrew	Shares	26000
Andrew	Sales	23000
Robert	Bonus	13000

Employees may have different sources of income and we want the following result:

Category	Source1	Income1	Source2	Income2
David	Salary	8000	Bonus	15000
Daniel	Salary	9000		
Andrew	Shares	26000	Sales	23000
Robert	Bonus	13000		

✦ 10.5 Row to column transposition with dynamic columns by filling into a table



We are not sure about the number of columns or even the names of columns after row to column conversion. In this case, we can't use the pivot function, but need to code the dynamic transposition.

The SPL script is as follows:

	A	B
1	<code>=connect("db")</code>	<code>=A1.query("select * from Income")</code>
2	<code>=B1.group(Name)</code>	<code>=A2.max(~.len())</code>
3	<code>=create(Name, \${B2.("Source"+string(~)+"", "Income"+string(~)).concat@c()})</code>	
4	<code>for A2</code>	<code>=A4. Name A4.conj([Source, Income])</code>
5		<code>>A3.record(B4)</code>

A3: Determine the number of columns according to the maximum number of members in a group after grouping, dynamically generate column names, and create a table sequence.

A4~B5: Loop through each group, concatenate the name, income sources and income amounts into a sequence, and add it to A3's table sequence.

✦ 10.5 Row to column transposition with dynamic columns by filling into a table



First generate target data structure for dynamic transposition

The key of dynamic transposition is to generate the target data structure first. After the table structure is determined, the data is put together into records according to the data structure and inserted into the target table. This idea is also applicable to some static transpositions.

✦ 10.6 Convert multiple rows to multiple rows of another form



Convert multiple rows to multiple rows of another form: Merge and calculate multiple records and generate multiple records in another form.

In the daily attendance information table, each card has 7 pieces of data every day. Get each employee's in and out information in each day.

Per_Code	in_out	Date	Time	Type
1110263	1	2013-10-11	09:17:14.0000000	In
1110263	6	2013-10-11	11:37:00.0000000	Break
1110263	5	2013-10-11	11:38:21.0000000	Return
1110263	0	2013-10-11	11:43:21.0000000	NULL
1110263	6	2013-10-11	13:21:30.0000000	Break
1110263	5	2013-10-11	14:25:58.0000000	Return
1110263	2	2013-10-11	18:28:55.0000000	Out

Every seven pieces of data are a group. We want to convert them into the following result:

Per_Code	Date	In	Out	Break	Return
1110263	2013-10-11	9:17:14	18:28:55	11:37:00	11:38:21
1110263	2013-10-11	9:17:14	18:28:55	13:21:30	14:25:58

✦ 10.6 Convert multiple rows to multiple rows of another form



Although the structure of the transposed table can be determined, it is very complex to implement it with pivot. We can create the target data structure first, and then fill in the data. The SPL script is as follows:

	A	B
1	<code>=connect("db").query("select * from DailyTime order by Per_Code,Date,Time")</code>	<code>=A1.group((#-1)\7)</code>
2	<code>=create(Per_Code,Date,In,Out,Break,Return)</code>	
3	<code>=B2.conj([~.Per_Code,~.Date] ~.(Time).m([1,7,2,3]) ~.Per_Code,~.Date] ~.(Time).m([1,7,5,6]))</code>	<code>>A2.record(A3)</code>

A1:Query and sort data by code, date and time.

B1:Every seven records are put into a group.

A2:Create result table sequence.

A3:For each group, retrieve records in the order of 1,7,2,3,1,7,5,6 to get the whole-day records. Organize values of each record into a sequence.

B3:Add values to the table sequence created by A2.

✦ 10.7 Transpose rows to columns by position-based value assignment



Row to column transposition by assigning values according to positions: Dynamically generate the data structure of the target table sequence, and directly assign values according to their positions.

According to the user records, dynamically generate user columns and get users' weekly online status. Then according to the related user table and record table, summarize the weekly user activities in 2018:



To display whether the user has a record of activity every week in 2018, as in the following table:

Week	User1	User2	User3
1	Yes	No	Yes
2	Yes	Yes	No
3	Yes	No	Yes
4	No	Yes	Yes

✦ 10.7 Transpose rows to columns by position-based value assignment



Create the target data structure first, and then fill in the data.

The SPL script is as follows:

	A	B
1	<code>=connect("db").query("select t1.ID as ID, t1.Name as Name, t2.Date as Date from User t1, Record t2 where t1.ID=t2.ID and year(t2.Date)=2018")</code>	
2	<code>=A1.derive(interval@w("2018-01-01",Date)+1:Week)</code>	<code>=A2.max(Week)</code>
3	<code>=A2.group(ID)</code>	<code>=B2.new(~:Week,\${A3.("\No\":"+Name).concat@c()})</code>
4	<code>=A3.run(~.run(B3(Week).field(A3.#+1,"Yes")))</code>	

A1: Query user table and record table, and join them by user ID.

A2: Calculate the week number according to the date, and store it in the new field *Week*.

B2: Find the largest week number.

A3: Group by user ID.

B3: Create an empty table sequence according to the maximum weekly number, and assign default values "No" to each day.

A4: For each piece of data in each group, locate the corresponding record in the target table through the weekly sequence number, and replace the user value with "Yes".

✦ 10.8 Transpose rows to columns, and do inter-column calculations at the same time



Row to column transposition with inter-column calculations at the same time.

Get the summary table of monthly due amount of each user in 2014 based on user payment detail table.

ID	customID	name	amount_payable	due_date	amount_paid	pay_date
112101	C013	CA	12800	2014/02/21	12800	2014/12/19
112102	C013	CA	3500	2014/06/15	3500	2014/12/15
112103	C013	CA	2600	2015/03/21		

Output the monthly payable amount in the specified year (such as 2014). If there is no data of the current month, the payable amount of the current month is the value of the previous month.

name	1	2	3	4	5	6	7	8	9	10	11	12
CA		12800	12800	12800	12800	16300	16300	16300	16300	16300	16300	16300
...												

✦ 10.8 Transpose rows to columns, and do inter-column calculations at the same time



Generate an empty result set first and then append data. The difference is that a series of calculations is needed to get the data to be appended. SPL script is as follows:

	A	B
1	=file("Payment.txt").import@t().select(year(due_date)==2014)	
2	=create(name,\${12.concat@c()})	=A1.group(customID)
3	for B2	=12.(null)
4		>A3.run(B3(month(due_date))= amount_payable)
5		>B3.run(~+=~[-1])
6		=A2.record(B2.name B3)

A1: Query records of 2014.

A2: Generate an empty result table sequence containing 12 months.

A3:Group by customID.

A3~B6:Loop through the groups, during which B4 sets the payable amount of the corresponding month.

B5 assigns null value to the previous month and perform the accumulation if there is new payable amount.

B6 inserts the record into the result table sequence.

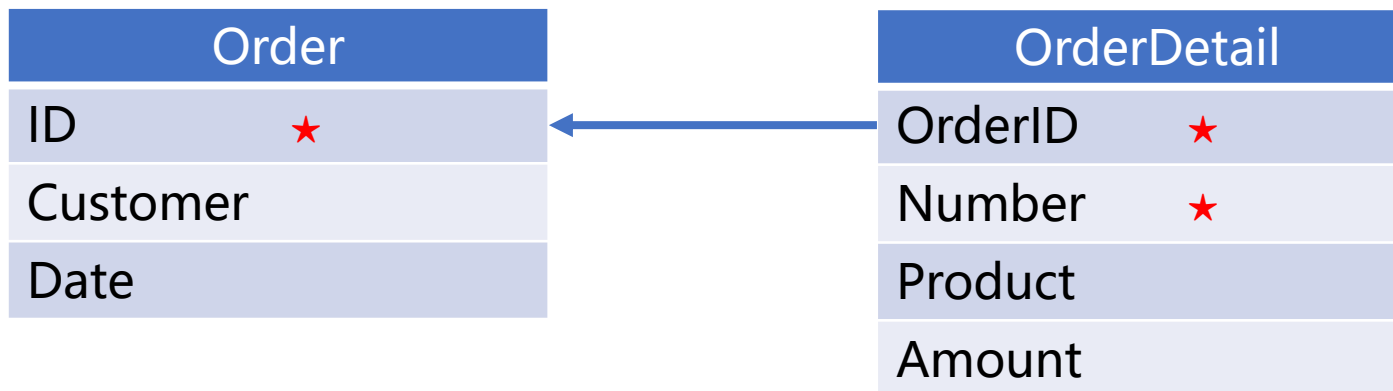
✦ 10.9 Dynamic transposition after the main and sub table join



Transpose rows into columns dynamically by inserting the sub table into the main table dynamically.

Based on the *order* table and *orderdetail* table, get the summary table of products purchased by each customer every day.

The relationship of *order* table and *orderdetail* table is that of the main and sub table. Each order record has multiple details records, as shown below:



The numbers of detail records in the *orderdetail* table corresponding to each order record are variable. Below is the desired table:

ID	Customer	Date	Product1	Amount1	Product2	Amount2	Product3	Amount3
1	3	20190101	Apple	5	Milk	3	Salt	1
2	5	20190102	Beef	2	Pork	4		
3	2	20190102	Pizza	3				

✦ 10.9 Dynamic transposition after the main and sub table join



The SPL script is as follows:

	A	B
1	<code>=connect("db") .query("select * from OrderDetail left join Order on Order.ID=OrderDetail.OrderID")</code>	
2	<code>=A1.group(ID)</code>	<code>=A2.max(~.count()).("Product"+string(~)+", "+"Amount"+string(~)).concat@c()</code>
3	<code>=create(ID, Customer, Date, \${B2})</code>	<code>>A2.run(A3.record([ID, Customer, Date] ~.([Product, Amount])).conj()))</code>

A1:Join the order table and the order detail table

A2:Group by order ID.

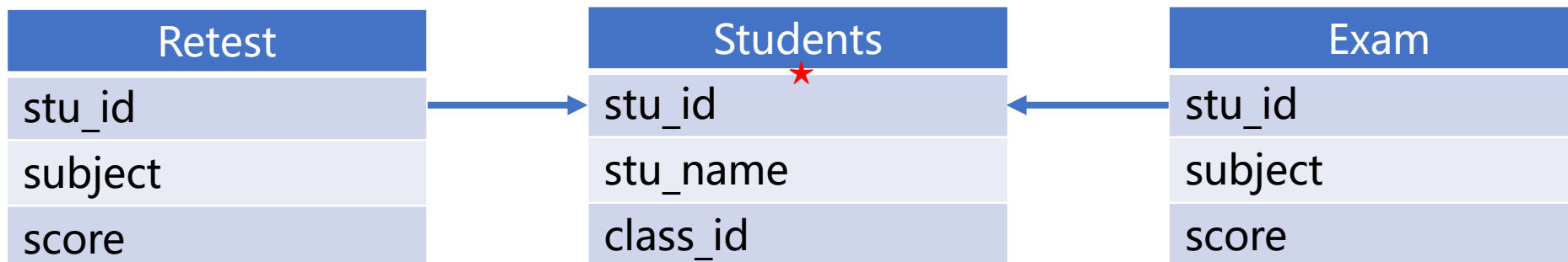
B2~A3:According to the maximum number of members in each group, dynamically generate column names and create an empty table sequence.

B3:Loop members in each group, dynamically put together the desired data and add them to the table sequence created by A3.

✦ 10.10 Dynamic row to column transposition after multi-table join



Perform dynamic row to column conversion based on the join of multiple related tables. Generate multiple columns dynamically according to field values, and then fill in the result of associated to them. Based on the *Students* table, *Exam* table and *Retest* table, get the summary table consisting of score of each subject, total score and retest score for each student.



The desired summary table is as follows:

stu_id	stu_name	Chinese_score	Math_score	total_score	Chinese_retest	Math_retest
1	Ashley	80	77	157		
2	Rachel	58	67	125	78	
3	Emily	85	56	141		82

✦ 10.10 Dynamic row to column transposition after multi-table join



The SPL script is as follows:

	A	B
1	<code>\$()select t1.stu_id stu_id,t1.stu_name stu_name,t2.subject subject,t2.score score1,t3.score score2 from Students.txt t1 left join Exam.txt t2 on t1.stu_id=t2.stu_id left join Retest.txt t3 on t1.stu_id=t3.stu_id and t2.subject=t3.subject order by t1.stu_id,t2.subject</code>	
2	<code>=A1.group(stu_id)</code>	<code>=A1.group(subject)</code>
3	<code>=create(stu_id,stu_name,\${(B2.(~.subject+"_score")) "total_score" B2.(~.subject+"_retest")}.concat@c())</code>	
4	<code>>A2.run(A3.record([stu_id,stu_name] B2.(~(A2.#).score1) A2.~.sum(score1) B2.(~(A2.#).score2)))</code>	

A1:Join the *Student* table, the *Exam* table and the *Retest* table, sort the result by student id and subject. The score of exam is score1 and the score of retest is score2.

A2,B2:Group by student id and subject respectively.

A3:According to the subjects, dynamically generate column names and create an empty table sequence.

A4:Loop members in student scores group, dynamically put together the desired data and add them to the table sequence created by A3.

✦ Summary of inter-table join transposition



Perform the join first for an inter-table join transposition

First, join data tables into a single table through associative relationship. The next steps are similar to the transpositions introduced earlier.

There are rich functions for doing calculations over SPL table sequences. They can meet most of the business computing requirements.

✦ 10.11 Transposition in column-layout



To present data in a column-layout is to rearrange a table from the vertical layout into a layout of two or more column groups.

Here is a world urban population table:

Continent	Country	City	Population
Africa	Egypt	Cairo	6789479
Asia	China	Shanghai	24240000
Europe	Britain	London	7285000
...

List the names and population of each city with a population of more than 2 million in Europe and Africa in a separate column group (each column group is sorted in descending order). The expected result is as follows:

Europe City	Population	Africa City	Population
Moscow	8389200	Cairo	6789479
London	7285000	Kinshasa	5064000
St Petersburg	4694000	Alexandria	3328196

✦ 10.11 Transposition in column-layout



The idea is to create the target data structure first, and then fill data in it.
The SPL script is as follows:

	A	B
1	<code>=connect("db").query("select * from World where Continent in('Europe', 'Africa') and Population >= 2000000")</code>	
2	<code>=A1.select(Continent:"Europe")</code>	<code>=A1.select(Continent:"Africa")</code>
3	<code>=create('Europe City',Population,'Africa City',Population)</code>	<code>=A3.paste(A2.(City),A2.(Population),B2.(City),B2.(Population))</code>

A1: Connect to the database and retrieve records of cities in Europe and Africa with more than 2 million population.

A2~B2: Get records of Europe and Africa respectively.

A3: Create an empty table sequence according to the target structure.

B3: Paste the sequence of values directly to the corresponding column using the table sequence's paste function.



Chapter 11

SPL COOKBOOK

Recursion

✦ 11.1 Recursively search single references



Recursively search for all levels of references of the specified field in the specified record after a table is joined with itself.

From the organizational structure table of a company, query all the superior organizations of Beijing market research team.

ID	ORG_NAME	PARENT_ID
1	Head Office	0
2	Beijing Branch Office	1
3	Shanghai Branch Office	1
4	Chengdu Branch Office	1
5	Beijing R&D Center	2
...

✦ 11.1 Recursively search single references



SPL is as follows, in which the rvs() function is used to reverse a sequence:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Organization")	/Query organization table
3	>A2.switch(PARENT_ID,A2:ID)	/Map the foreign key Parent_ID to the corresponding record to perform a self-join
4	=A2.select@1(ORG_NAME=="Beijing Market Research Team")	/Select the record where Beijing market research team is located
5	=A4.prior(PARENT_ID)	/Use the prior function to find the superior organizations
6	=A5.rvs().(ORG_NAME).concat(" / ")	/Use rvs function to arrange the superior organizations top-down

A6

Value

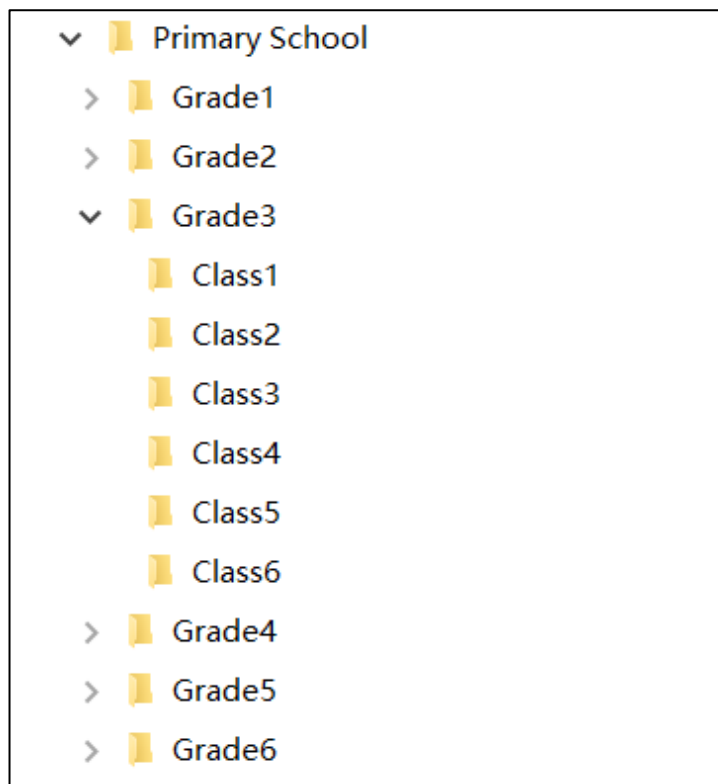
Head Office / Beijing Branch Office / Beijing Marketing Department / Beijing Market Research Team

✦ 11.2 Traverse all files in the directory



Recursively traverse all the files in the specified directory.

A primary school investigates the terminals students use in online learning. Count the proportion of each type of terminal.



ID	STUDENT_NAME	TERMINAL
1	Rebecca Moore	Phone
2	Ashley Wilson	Phone,PC,Pad
3	Rachel Johnson	Phone,PC,Pad
4	Emily Smith	PC,Pad
5	Ashley Smith	PC
6	Matthew Johnson	Phone
7	Alexis Smith	Phone,PC
8	Megan Wilson	Phone,PC,Pad
...

✦ 11.2 Traverse all files in the directory



SPL is as follows, in which the `directory@s()` function is used to recursively search for files:

	A	B	C
1	<code>=directory@ps("D:/Primary School")</code>		/Recursively traversing directories, to list all files
2	<code>>totalCount=0</code>		/Define a variable totalCount to store the total number of records
3	for A1	<code>=file(A3).xlsimport@t()</code>	/Import the Excel file of each class's questionnaire in loop
4		<code>=B3.conj(TERMINAL.split@c()) B4</code>	/Split terminals by commas and merge them into B4's sequence
5		<code>>totalCount+=B3.len()</code>	/Add the number of students in each class to totalCount
6	<code>=B4.groups(~:TERMINAL;count(~)/totalCount:PERCENTAGE)</code>		/Group and summarize B4's sequence and calculate the percentage of each type of terminal

A1		B4		A6	
Member		Member		TERMINAL	PERCENTAGE
D:\Primary School\Grade1\Class1\Investigation.xlsx		Phone	groups →	PC	0.7
D:\Primary School\Grade1\Class2\Investigation.xlsx		Phone		Pad	0.567
D:\Primary School\Grade1\Class3\Investigation.xlsx		PC		Phone	0.933
...					

✦ 11.3 Recursively search all references by loop



Recursively search for all levels of references of the specified field in all records after a table is joined with itself.

In the organization table of a company, query the level of each department.

ID	ORG_NAME	PARENT_ID
1	Head Office	0
2	Beijing Branch Office	1
3	Shanghai Branch Office	1
4	Chengdu Branch Office	1
5	Beijing R&D Center	2
...

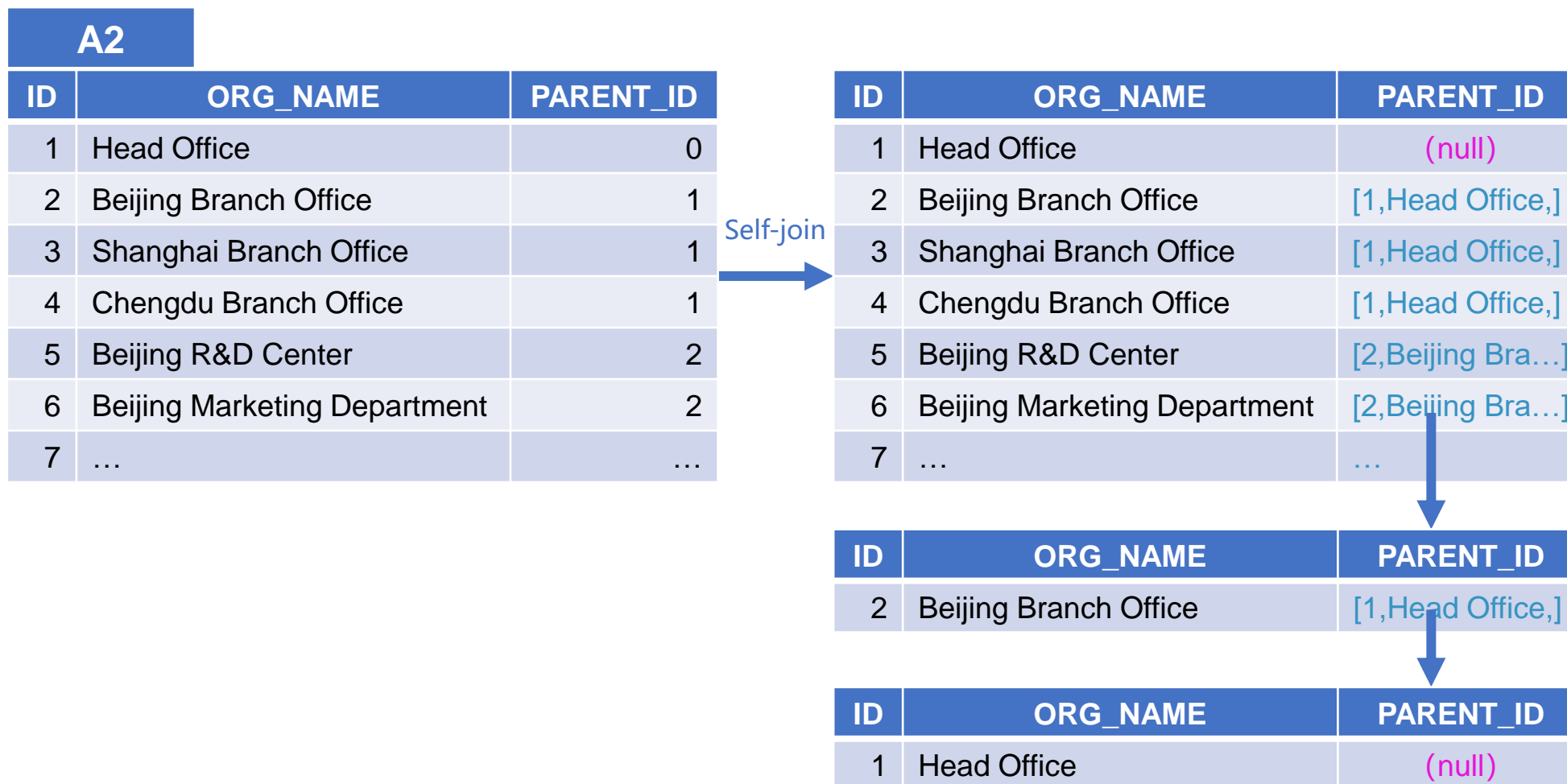
✦ 11.3 Recursively search all references by loop



SPL is as follows, in which the prior function is used to recursively search references:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Organization")	/Query organization table
3	>A2.switch(PARENT_ID,A2:ID)	/Map foreign key Parent_ID to the corresponding record to perform a self-join
4	=A2.new(ID,ORG_NAME,~.prior(PARENT_ID).len()-1:LEVEL)	/Create a new table consisting of ID, department names, and levels. The level is calculated by recursively finding the number of levels of the referenced records through the prior function.

✦ 11.3 Recursively search all references by loop



✦ 11.3 Recursively search all references by loop



A4

ID	ORG_NAME	~.prior(PARENT_ID)
1	Head Office	(null)
2	Beijing Branch Office	[1,Head Office,]
3	Shanghai Branch Office	[1,Head Office,]
4	Chengdu Branch Office	[1,Head Office,]
5	Beijing R&D Center	[2,Beijing Branch O...]
6	Beijing Marketing Department	[2,Beijing Branch O...]
7



ID	ORG_NAME	LEVEL
1	Head Office	0
2	Beijing Branch Office	1
3	Shanghai Branch Office	1
4	Chengdu Branch Office	1
5	Beijing R&D Center	2
6	Beijing Marketing Department	2
7



ID	ORG_NAME	PARENT_ID
6	Beijing Marketing Department	[2,Beijing Branch O...]
2	Beijing Branch Office	[1,Head Office,]
1	Head Office	(null)

✦ 11.4 Recursively search references until the specified value



Search all levels of references recursively until the specified value after a table is joined with itself.
In the organization table of a company, query the subordinate organizations of Beijing Branch and list the superior organizations of each of them.

ID	ORG_NAME	PARENT_ID
1	Head Office	0
2	Beijing Branch Office	1
3	Shanghai Branch Office	1
4	Chengdu Branch Office	1
5	Beijing R&D Center	2
...

✧ 11.4 Recursively search references until the specified value



SPL is as follows, in which the `prior(F,r')` function is used to recursively search references until the specified value:

	A	B
1	<code>=connect("db")</code>	/Connect to database
2	<code>=A1.query("select * from Organization")</code>	/Query organization table
3	<code>>A2.switch(PARENT_ID,A2:ID)</code>	/Map foreign key Parent_ID to the corresponding record to perform a self-join
4	<code>=A2.select@1(ORG_NAME=="Beijing Branch Office")</code>	/Select the record of Beijing Branch
5	<code>=A2.new(ID,ORG_NAME,~.prior(PARENT_ID,A4):PARENT)</code>	/Create a new table consisting of ID, department name, and parent. The parent is obtained by recursively searching for the records under Beijing branch through prior function.
6	<code>=A5.select(PARENT!=null)</code>	/Select the members whose parent exists, otherwise they are not subordinates of Beijing Branch.
7	<code>=A6.run(PARENT=PARENT.(PARENT_ID.ORG_NAME).concat@c())</code>	/Concatenate all the parent names in the parent field, separated by commas.

✦ 11.4 Recursively search references until the specified value



A5		
ID	ORG_NAME	PARENT
1	Head Office	(null)
2	Beijing Branch Office	[]
3	Shanghai Branch Office	(null)
4	Chengdu Branch Office	(null)
5	Beijing R&D Center	[[5,Beijing R&D Center,]]
6	Beijing Marketing Department	[[5,Beijing Marketing Department,]]
7	Beijing AI R&D Department	[[7,Beijing AI R&D Department,],[5,Beijing Marketing Department,]]
8	Beijing Internet R&D Department	[[8, Beijing Internet R&D Department,],[5,Beijing R&D Center,]]
9



ID	ORG_NAME	PARENT_ID
8	Beijing Internet R&D Department	[5,Beijing R&D Center,]
5	Beijing R&D Center	[2,Beijing Branch Office,]

✦ 11.4 Recursively search references until the specified value



A6	ID	ORG_NAME	PARENT
	2	Beijing Branch Office	[]
	5	Beijing R&D Center	[[5,Beijing R&D Center,]]
	6	Beijing Marketing Department	[[5,Beijing Marketing Department,]]
	7	Beijing AI R&D Department	[[7,Beijing AI R&D Department,],[5,Beijing Marketing Department,]]
	8	Beijing Internet R&D Department	[[8, Beijing Internet R&D Department,],[5,Beijing R&D Center,]]
	9

A7	ID	ORG_NAME	PARENT
	2	Beijing Branch Office	
	5	Beijing R&D Center	Beijing Branch Office
	6	Beijing Marketing Department	Beijing Branch Office
	7	Beijing AI R&D Department	Beijing R&D Center,Beijing Branch Office
	8	Beijing Internet R&D Department	Beijing R&D Center,Beijing Branch Office
	9	Beijing Internet Interface R&D department	Beijing Internet R&D Department,Beijing R&D Center,Beijing Branch Office
	10	Beijing Market Research Team	Beijing Marketing Department,Beijing Branch Office

✦ 11.5 Search the upper level reference



Only search for the upper level reference of a record after a table is joined with itself.
In the Chinese administrative division table, query the name of the superior region of each administrative region.

ID	NAME	PARENT_ID
1	China	0
11	Beijing	1
12	Tianjin	1
13	Hebei	1
...
1301	Shijiazhuang	13
1302	Tangshan	13
...

✦ 11.5 Search the upper level reference



SPL is as follows, in which the P.nodes(F) function is used to find the upper level reference of a record:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from ChinaRegion")	/Query ChinaRegion table
3	>A2.switch(PARENT_ID,A2:ID)	/Map foreign key Parent_ID to the corresponding record to perform a self-join
4	=A2.nodes(PARENT_ID)	/Use the nodes function to find the upper level reference of a record
5	=A4.new(ID,NAME,PARENT_ID.NAME:PARENT_NAME)	/Create a table sequence consisting of ID, name and upper level region name

✦ 11.5 Search the upper level reference



A4		
ID	NAME	PARENT_ID
1	China	(null)
11	Beijing	[1,China,]
12	Tianjin	[1,China,]
13	Hebei	[1,China,]
...
1301	Shijiazhuang	[13,Hebei,]
1302	Tangshan	[13,Hebei,]
1303	Qinhuangdao	[13,Hebei,]
...
130102	Changan District	[1301,Shijiazhuang,]
130104	Qiaoxi District	[1301,Shijiazhuang,]
130105	Xinhua District	[1301,Shijiazhuang,]
...



A5		
ID	NAME	PARENT_NAME
1	China	(null)
11	Beijing	China
12	Tianjin	China
13	Hebei	China
...
1301	Shijiazhuang	Hebei
1302	Tangshan	Hebei
1303	Qinhuangdao	Hebei
...
130102	Changan District	Shijiazhuang
130104	Qiaoxi District	Shijiazhuang
130105	Xinhua District	Shijiazhuang
...

✦ 11.6 Find records with the specified value in the reference chain with the parent value listed



Get records with specified value in the reference chain and list the upper level value.

In the Chinese administrative division table, query the subordinate administrative regions of Hebei Province.

ID	NAME	PARENT_ID
1	China	0
11	Beijing	1
12	Tianjin	1
13	Hebei	1
...
1301	Shijiazhuang	13
1302	Tangshan	13
...

✦ 11.6 Find records with the specified value in the reference chain with the parent value listed



SPL is as follows, in which the P.nodes(F,r) function is used to get the records with the specified value in the reference chain:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from ChinaRegion")	/Query ChinaRegion table
3	>A2.switch(PARENT_ID,A2:ID)	/Map foreign key Parent_ID to the corresponding record to perform a self-join
4	=A2.select@1(name=="Hebei")	/Select the record of Hebei province
5	=A2.nodes(PARENT_ID,A4)	/Use the nodes function to recursively search references until PARENT_ID points to Hebei Province
6	=A5.new(ID,NAME,PARENT_ID.NAME:PARENT_NAME)	/Create a table sequence consisting of ID, name and upper level region name.

✦ 11.6 Find records with the specified value in the reference chain with the parent value listed



A5		
ID	NAME	PARENT_ID
1301	Shijiazhuang	[13,Hebei,]
1302	Tangshan	[13,Hebei,]
1303	Qinhuangdao	[13,Hebei,]
1304	Handan	[13,Hebei,]
1305	Xingtai	[13,Hebei,]
...
130102	Changan District	[1301,Shijiazhuang,]
130104	Qiaoxi District	[1301,Shijiazhuang,]
130105	Xinhua District	[1301,Shijiazhuang,]
130107	Jingxing mining area	[1301,Shijiazhuang,]
130108	Yuhua District	[1301,Shijiazhuang,]
130109	Gaocheng District	[1301,Shijiazhuang,]
...



A6		
ID	NAME	PARENT_NAME
1301	Shijiazhuang	Hebei
1302	Tangshan	Hebei
1303	Qinhuangdao	Hebei
1304	Handan	Hebei
1305	Xingtai	Hebei
...
130102	Changan District	Shijiazhuang
130104	Qiaoxi District	Shijiazhuang
130105	Xinhua District	Shijiazhuang
130107	Jingxing mining area	Shijiazhuang
130108	Yuhua District	Shijiazhuang
130109	Gaocheng District	Shijiazhuang
...

✦ 11.7 Search for leaf records



Only search all leaves (records not referenced by other records) after a table is joined with itself.

In the Chinese administrative division table, query the subordinate districts and counties of Hebei Province.

ID	NAME	PARENT_ID
1	China	0
11	Beijing	1
12	Tianjin	1
13	Hebei	1
...
1301	Shijiazhuang	13
1302	Tangshan	13
...



SPL is as follows, in which the P.nodes@d(F,r) function is used to recursively search all leaves:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from ChinaRegion")	/Query ChinaRegion table
3	>A2.switch(PARENT_ID,A2:ID)	/Map foreign key Parent_ID to the corresponding record to perform a self-join
4	=A2.select@1(name=="Hebei")	/Select the record of Hebei province
5	=A2.nodes@d(PARENT_ID,A4)	/Use the nodes function to recursively search references until PARENT_ID points to Hebei Province; the @d option gets leaves only. In this case, all districts and counties will be selected
6	=A5.new(ID,NAME,PARENT_ID.NAME:PARENT_NAME)	/Create a table sequence consisting of ID, name and upper level region name.

✦ 11.7 Search for leaf records



A5		
ID	NAME	PARENT_ID
130102	Changan District	[1301,Shijiazhuang,]
130104	Qiaoxi District	[1301,Shijiazhuang,]
130105	Xinhua District	[1301,Shijiazhuang,]
130107	Jingxing mining area	[1301,Shijiazhuang,]
130108	Yuhua District	[1301,Shijiazhuang,]
130109	Gaocheng District	[1301,Shijiazhuang,]
130110	Luquan District	[1301,Shijiazhuang,]
130111	Luancheng District	[1301,Shijiazhuang,]
130121	Jingxing County	[1301,Shijiazhuang,]
130123	Zhengding County	[1301,Shijiazhuang,]
130125	Xingtang County	[1301,Shijiazhuang,]
130621	Lingshou County	[1301,Shijiazhuang,]
...



A6		
ID	NAME	PARENT_NAME
130102	Changan District	Shijiazhuang
130104	Qiaoxi District	Shijiazhuang
130105	Xinhua District	Shijiazhuang
130107	Jingxing mining area	Shijiazhuang
130108	Yuhua District	Shijiazhuang
130109	Gaocheng District	Shijiazhuang
130110	Luquan District	Shijiazhuang
130111	Luancheng District	Shijiazhuang
130121	Jingxing County	Shijiazhuang
130123	Zhengding County	Shijiazhuang
130125	Xingtang County	Shijiazhuang
130621	Lingshou County	Shijiazhuang
...

✦ 11.8 Find all upper level references



Find all upper level references after a table is joined with itself.

In the Chinese administrative division table, list all the superior regions of each administrative region.

The output for Shijiazhuang, for instance, is China, Hebei, Shijiazhuang.

ID	NAME	PARENT_ID
1	China	0
11	Beijing	1
12	Tianjin	1
13	Hebei	1
...
1301	Shijiazhuang	13
1302	Tangshan	13
...

✦ 11.8 Find all upper level references



SPL is as follows, in which the `P.nodes@p(F,r)` function is used to find all upper level references:

	A	B
1	<code>=connect("db")</code>	/Connect to database
2	<code>=A1.query("select * from ChinaRegion")</code>	/Query ChinaRegion table
3	<code>>A2.switch(PARENT_ID,A2:ID)</code>	/Map foreign key Parent_ID to the corresponding record to perform a self-join
4	<code>=A2.nodes@p(PARENT_ID)</code>	/Use @p option with the nodes function to recursively find all upper level references
5	<code>=A4.run(~=~.(NAME).concat@c())</code>	/Concatenate names of the upper level references, separated by commas

✦ 11.8 Find all upper level references



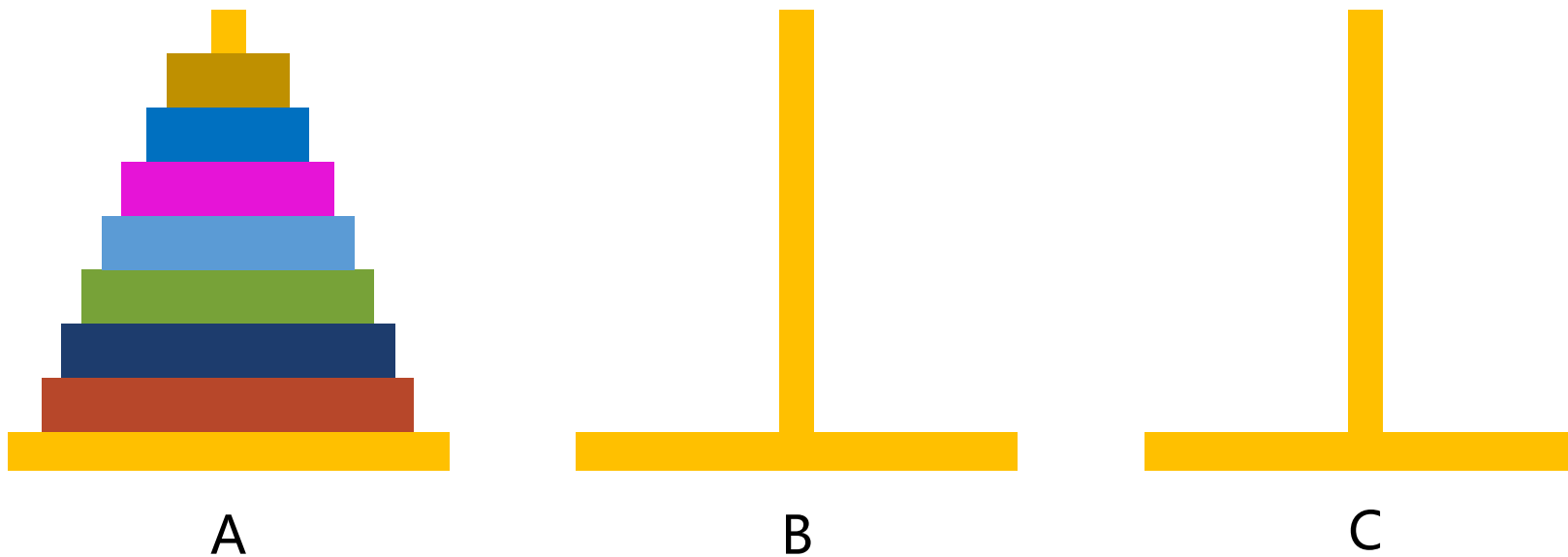
A4		A5
Member		Member
[[1,China,]]		China
[[1,China,],[11,Beijing,]]		China,Beijing
[[1,China,],[12,Tianjin,]]		China,Tianjin
[[1,China,],[13,Hebei,]]		China,Hebei
...		...
[[1,China,],[13,Hebei,],[1301,Shijiazhuang,]]	→	China,Hebei,Shijiazhuang
[[1,China,],[13,Hebei,],[1302,Tangshan,]]		China,Hebei,Tangshan
[[1,China,],[13,Hebei,],[1303,Qinhuangdao]]		China,Hebei,Qinhuangdao
...		...
[[1,China,],[13,Hebei,],[1301,Shijiazhuang,],...]		China,Hebei,Shijiazhuang,Changan District
[[1,China,],[13,Hebei,],[1301,Shijiazhuang,],...]		China,Hebei,Shijiazhuang,Qiaoxi District
[[1,China,],[13,Hebei,],[1301,Shijiazhuang,],...]		China,Hebei,Shijiazhuang,Xinhua District
...		...

✦ 11.9 Hanoi Tower problem



Recursively call function to solve Hanoi Tower problem.

The Hanoi Tower problem is a classical recursive problem. The objective of the puzzle is to move all the disks on rod A to rod C, with the original order kept. Only one disk can be moved at a time, and always keep a smaller disk on top of a larger one.



✦ 11.9 Hanoi Tower problem



The disks are named 1 to n from small to large. We always treat the n disks as two groups: the nth disk and the other n-1 ones. Move n-1 disks to rod B, the nth disk to rod C, and then the n-1 disks to rod C. The SPL is as follows, where func(c,...) is used to perform the recursive operation:

	A	B	C	D
1	func			/Define a function
2		if(A1==1)	>output("move disk " + string(A1) + " from " + B1 + " to " + D1)	/Move it to C when there is only one disk
3		else	>func(A1,A1-1,B1,D1,C1)	/Move n-1 disks on A to B
4			>output("move disk " + string(A1) + " from " + B1 + " to " + D1)	/Move the bottom disk on A to C
5			>func(A1,A1-1,C1,B1,D1)	/Move n-1 disks on B to A
6	>func(A1,5,"A","B","C")			/The function has four parameters: number of disks (also the names), initial rod, intermediate rod and target rod

✦ 11.9 Hanoi Tower problem



When there are 5 disks on rod A,
the output result is as shown on
the right:

```
move disk 1 from A to C
move disk 2 from A to B
move disk 1 from C to B
move disk 3 from A to C
move disk 1 from B to A
move disk 2 from B to C
move disk 1 from A to C
move disk 4 from A to B
move disk 1 from C to B
move disk 2 from C to A
move disk 1 from B to A
move disk 3 from C to B
move disk 1 from A to C
move disk 2 from A to B
move disk 1 from C to B
move disk 5 from A to C
move disk 1 from B to A
move disk 2 from B to C
move disk 1 from A to C
```

```
move disk 3 from B to A
move disk 1 from C to B
move disk 2 from C to A
move disk 1 from B to A
move disk 4 from B to C
move disk 1 from A to C
move disk 2 from A to B
move disk 1 from C to B
move disk 3 from A to C
move disk 1 from B to A
move disk 2 from B to C
move disk 1 from A to C
```

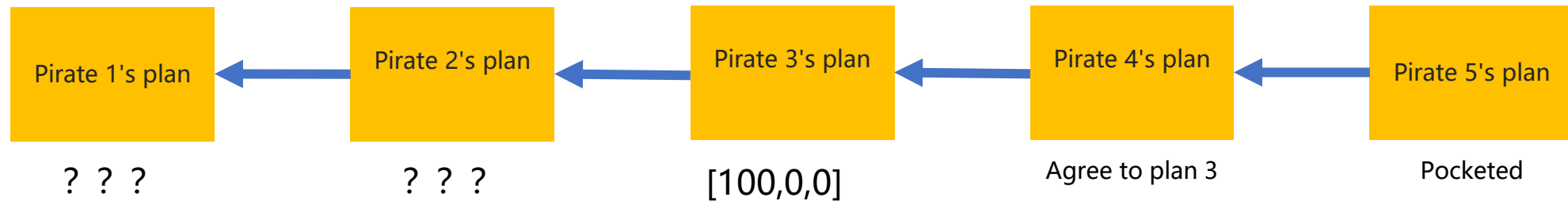
✦ 11.10 Pirate treasure division problem



Recursively call function to solve the pirate treasure division problem.

The puzzle is like this:

Five pirates snatched 100 gold coins. They propose plans of dividing the coins by drawing lots: first, No. 1 put forwards the distribution plan, and then five people vote. More than half of the votes is needed to approve the plan, or he will be thrown into the sea to feed sharks, and so on.



When there are only two pirates, pirate 5 will veto any plan pirate 4 proposes and pocket 100 gold coins. So in order to protect his life, pirate 4 will unconditionally agree with pirate 3. Knowing the idea of pirate 4, the greedy pirate 3 will surely give the [100,0,0] distribution plan. This logic also applies to any of the other pirates.

✦ 11.10 Pirate treasure division problem



The SPL is as follows, where func(c,...) is used to perform the recursive operation:

	A	B	C	D	E
1	func				
2		if(A1==2)	return [-1,B1]	/When there are only two pirates	/The last pirate takes all
3		=func(A1,A1-1,B1)	/Use func() function to recursively calculate the plan that the other pirates will adopt after I am rejected		
4		=B3.psort()	=A1/2	/Sort the next plans	/How many pirates are needed to support me
5		for B4.len()	if (B5 <= C4)	=B3(B4(B5)) + = 1	/Strive for the support of the pirates with the least distribution and give 1 more coin
6				> B1 -= D5	/Deduct the allocated gold coins from the total, and the rest are mine
7			else	> B3(B4(B5))=0	/Other pirates are allocated none
8		return B1 B3	/Return my remaining gold coins and the next modified plan, which is my allocation plan.		
9		=func(A1,5,100)	/Execute func() function, where the parameters are 5 pirates and 100 gold coins		

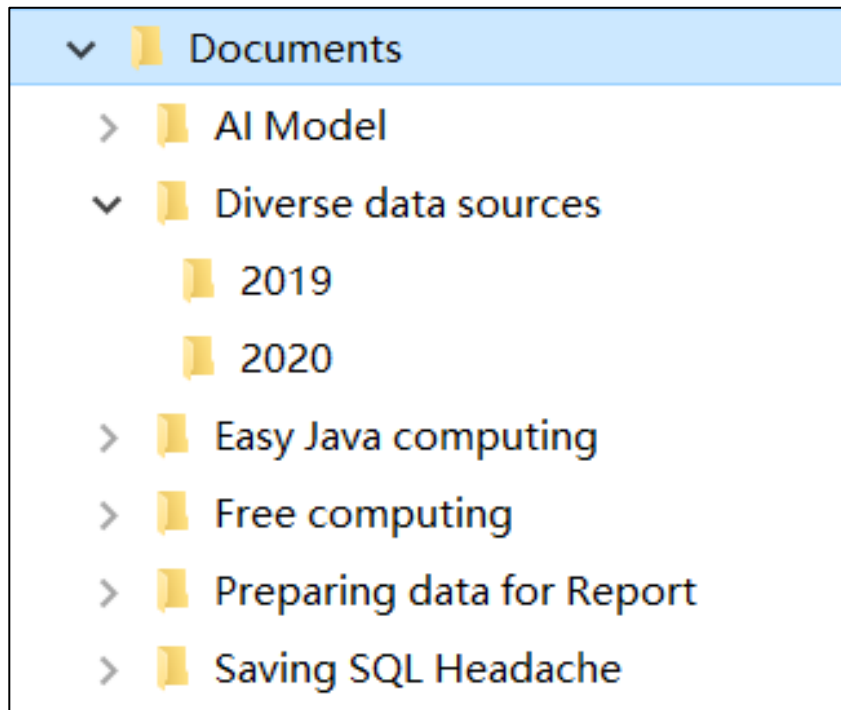
A9	Member
	97
	0
	1
	2
	0

✦ 11.11 Traverse the directories to summarize all the files



Traverse the directories and recursively call the script to summarize all the files.

Traverse all text files in the specified directory and its subdirectories, and append the 17th line in each file to a target file.



Sample of the text file:s

```
16 ...  
17 Middleware for report source data computing  
18 ...
```

Result.txt after summary:

```
Middleware for report source data computing  
SPL parsing and exporting Excel  
SQL Headaches Therapies – For Loop Operations  
The skill of updating database with esProc  
...
```

✦ 11.11 Traverse the directories to summarize all the files



First define a parameter:

ID	Name	Value	Remarks
1	path		File directory

The following script is used to read the specified line of each of the files and export it to a text file. It is saved as *readfile.dfx*. SPL is as follows:

	A	B	C
1	=directory@p(path)		/List files in the current directory
2	=A1.(file(~).cursor@s())		/Open file cursor in loop
3	=A2.((~.skip(16),~.fetch@x(1)))		/Skip 16 lines in each file cursor to get line 17
4	=A3.union()		/Union the fetched records
5	>file("result.txt").export@a(A4)		/Append the results to the <i>result.txt</i> file
6	=directory@dp(path)		/List subdirectories in the current directory
7	if A6.len()==0	return	/If there is no subdirectory, the program returns result
8	else	=A6.(call("readfile.dfx",~))	/If there are subdirectories, execute the script recursively

Use call() function to execute the edited *readfile.dfx*. SPL is as follows:

	A	B
1	=call("readfile.dfx","D:/Documents")	/Execute the cellset program by specifying the directory with parameter

The recursive traversal of directories and cursor-type file reading almost uses no memory. It is suitable for processing a large number of files and big data.



Chapter 12

SPL
COOKBOOK

Using structured text data

✦ 12.1 Filter small files



Get records that meet the specified condition in a text file.
Find scores of students in class 10 in *students_scores.txt*.

Text file content:

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
1	Allen Ashley	98	97	97
1	Allen Brandon	93	76	78
.....				

✦ 12.1 Filter small files



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores.txt").import @t()</code>	<code>/@t</code> option reads the first line as the title; the separator by default is <code>"\t"</code>
2	<code>=A1.select(CLASS==10)</code>	<code>/Select</code> records of students in class 10, which is calculated immediately

A2's result:

CLASS	Name	English	Chinese	Math
10	Adams Ashley	89	49	91
10	Adams Kayla	85	74	45
10	Allen Danielle	62	77	88
...

✦ 12.2 Read certain fields in a text file



Read data of certain fields from a text file.

File content:

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
1	Allen Ashley	98	97	97
1	Allen Brandon	93	76	78
.....				

✦ 12.2 Read certain fields in a text file



SPL code

	A
1	=file(path).import@t(CLASS,Chinese)
2	=file(path).import@t(#1,#4)

SPL output

CLASS	Chinese
1	31
1	85
1	87
...	...

✦ 12.3 Read data in a text file using specified separator



From a text file with separator specified, read structured data using the specified separator.

File content: CLASS,NAME,English,Chinese,Math
1,Adams Brooke,63,31,69
1,Adams Hannah,89,85,79
1,Adams Jonathan,88,87,91
...

Separated by ","

CLASS|NAME|English|Chinese|Math
1|Adams Brooke|63|31|69
1|Adams Hannah|89|85|79
1|Adams Jonathan|88|87|91
...

Separated by "|"

✦ 12.3 Read data in a text file using specified separator



SPL code

	A
1	<code>=file(path).import@t(;;",")</code>
2	<code>=file(path).import@tc()</code>

	A
1	<code>=file(path).import@t(;;" ")</code>

SPL output

CLASS	Name	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...

✦ 12.4 Aggregate data in a small file to get sum



Perform SUM aggregate over a small file.

Calculate the total score of Chinese based on *students_scores.csv*.

Text file content:

```
CLASS,NAME,English,Chinese,Math
1,Adams Brooke,63,31,69
1,Adams Hannah 89,85,79
1,Adams Jonathan,88,87,91
1,Allen Ashley,98,97,97
1,Allen Brandon,93,76,78
.....
```

✦ 12.4 Aggregate data in a small file to get sum



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores.csv").import@t(;"", ")</code>	/Specify ";" as the separator
2	<code>=A1.sum(Chinese)</code>	/Get total of Chinese scores

A2's result

Value
181025

✦ 12.5 Inter-column calculation in a small file



Perform inter-column calculation in a text file.

Calculate the total score of each student based on *students_scores.txt*.

Text file content:

```
CLASS|NAME|English|Chinese|Math
1|Adams Brooke|63|31|69
1|Adams Hannah|89|85|79
1|Adams Jonathan|88|87|91
...
```


✦ 12.5 Inter-column calculation in a small file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores.txt").import@t(;" ")</code>	/Specify " " as the separator
2	<code>=A1.derive(English+Chinese+Math:total_score)</code>	/Add a column of total score

A2's result

CLASS	Name	English	Chinese	Math	total_score
1	Adams Brooke	63	31	69	163
1	Adams Hannah	89	85	79	253
1	Adams Jonathan	88	87	91	266
...

✦ 12.6 Perform comprehensive calculations using small text files



Perform comprehensive calculations using text file data.

Calculate the average Chinese score of all students and that of the students who passed in Chinese in class 10 based on *students_scores.txt*.

Text file content:

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...				

✦ 12.6 Perform comprehensive calculations using small text files



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores.txt").import@t(CLASS,Chinese;," ")</code>	/Read <i>Class</i> field and <i>Chinese</i> field
2	<code>=A1.select(CLASS==10)</code>	/Select records of Class 10
3	<code>=A2.avg(Chinese),A2.avg(if(Chinese>=60,Chinese))]</code>	/Calculate the two types of average

A3

Member
62.667
78.706

✦ 12.7 Read untitled structured text data



Read structured data in an untitled text file.

File content:

1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
1	Allen Ashley	98	97	97
1	Allen Brandon	93	76	78
1	Baker Danielle	83	40	95
.....				

✦ 12.7 Read untitled structured text data



SPL code

There is no title

	A
1	=file(path).import()

SPL output

_1	_2	_3	_4	_5
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...

✦ 12.8 Read a text file using specified data type and format



Specify the data type and format to read a text file. For example, using the non-default date format.

File content:

```
user_id,gender,age,insertdate
483833,M,19,2018/12/11
156772,M,31,2018/2/13
173388,M,34,2018/8/21
...
```

✦ 12.8 Read a text file using specified data type and format



SPL code

Data type of user_id: string; Date format: yyyy/M/d

	A	Read normally, and then modify with run function
1	=file(path).import@t()	
2	=A2.run(user_id=string(user_id),insertdate=date(insertdate,"yyyy/M/d"))	

	A
1	=file(path).import@t(user_id:string,gender,age,insertdate:date:"yyyy/M/d")

The correct way of reading a file in SPL

user_id	gender	age	insertdate
483833	M	19	2018/12/11
156772	M	31	2018/2/13
173388	M	34	2018/8/21
...

Read normally

SPL output

user_id	gender	age	insertdate
483833	M	19	2018-12-11
156772	M	31	2018-02-13
173388	M	34	2018-08-21
...	...		

Modify with run function

user_id	gender	age	insertdate
483833	M	19	2018-12-11
156772	M	31	2018-02-13
173388	M	34	2018-08-21
...

Read correctly

✦ 12.9 Read structured text data according to the specified character set



Read structured data in a text file according to the specified character set.

File content:

```
user_id,reg_mon,gender,age,cell_province,id_province,id_city,insertdate
483833,2017-04,男,19,c29,c26,c26241,2018-12-11
156772,2016-05,男,31,c11,c11,c11159,2018-02-13
173388,2016-05,男,34,c02,c02,c02182,2018-08-21
...
```


✦ 12.9 Read structured text data according to the specified character set



SPL code

	A	
1	=file(path).import@tc()	Normal reading

	A	
1	=file(path:"utf-8").import@tc()	Reading with specified character set

SPL output

user_id	reg_mon	gender	age	cell_province	id_province	id_city	insertdate	
483833	2017-04	□□	19	c29	c26	c26241	2018-12-11	
156772	2016-05	□□	31	c11	c11	c11159	2018-02-13	
173388	2016-05	□□	34	c02	c02	c02182	2018-02-13	
...	Normal reading

user_id	reg_mon	gender	age	cell_province	id_province	id_city	insertdate	
483833	2017-04	男	19	c29	c26	c26241	2018-12-11	
156772	2016-05	男	31	c11	c11	c11159	2018-02-13	
173388	2016-05	男	34	c02	c02	c02182	2018-02-13	
...	Reading with specified character set

✦ 12.10 Sort data in a small text file in ascending order



Sort the structured data in a text file by the values of a certain field in ascending order.
Sort records by Chinese score in ascending order based on *students_score.txt*.

File content:

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
1	Allen Ashley	98	97	97
1	Allen Brandon	93	76	78
.....				

✦ 12.10 Sort data in a small text file in ascending order



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_score.txt").import@t()</code>	/Read file
2	<code>=A1.sort(Chinese)</code>	/Sort by Chinese in ascending order

A2's result

Name	Math	Chinese	English
Hannah	90	76	95
Tyler	87	78	93
Zachary	75	81	85
...

✦ 12.11 Sort data in a small text file in descending order



Sort the structured data in a text file in descending order.

Sort records by total score in descending order base on *students_score.txt*.

Text file content:

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
1	Allen Ashley	98	97	97
1	Allen Brandon	93	76	78
.....				

✦ 12.11 Sort data in a small text file in descending order



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_score.txt").import@t()</code>	/Read file
2	<code>=A1.sort@z(Math+English+Chinese)</code>	/Sort by total score in descending order

A2's result

Name	Math	Chinese	English
Allen Ashley	98	97	97
Lewis Antony	93	92	94
Adams Jonathan	88	87	91
...

✦ 12.12 Sort structured data in a small text file by multi fields in specified order



Sort structured data in a text file by multiple fields in specified order.

Sort records by class in ascending order and by total score in descending order based on *students_scores.txt*.

Text file content:

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
1	Allen Ashley	98	97	97
1	Allen Brandon	93	76	78
.....				

✦ 12.12 Sort structured data in a small text file by multi fields in specified order



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores.txt").import@t()</code>	/Read file
2	<code>=A1.sort(CLASS,-(English+Chinese+Math))</code>	/Sort records by class in ascending order, and by total score in descending order

A2's result

Name	Math	Chinese	English
Allen Ashley	98	97	97
Lewis Antony	93	92	94
Adams Jonathan	88	87	91
...

✦ 12.13 Perform grouping & aggregation over a small file



Perform grouping & aggregation over records in a small text file.
Count the total number of logins in each province based on *user_info_reg.csv*.

Data in the file:

user_id	reg_mon	age	cell_province	id_province	id_city	insertdate	reg_time
483833	2017-04	19	c29	c26	c26241	2018-12-11	56558
156772	2016-05	31	c11	c11	c11159	2018-02-13	81617
...

✦ 12.13 Perform grouping & aggregation over a small file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/user_info_reg.csv").import@tc()</code>	/Read file
2	<code>=A1.groups(id_province;count(~):cnt)</code>	/Perform count after grouping

A2's result

id_province	cnt
c01	27202
c02	61735
c03	14433
...	...

✦ 12.14 Perform filter after grouping over a small file



Perform filtering after records in a text file are grouped.

Find the users whose total login time is less than 1000 minutes based on *user_info_reg.csv*.

Data in the file:

user_id	reg_mon	age	cell_province	id_province	id_city	insertdate	reg_time
483833	2017-04	19	c29	c26	c26241	2018-12-11	56558
156772	2016-05	31	c11	c11	c11159	2018-02-13	81617
...

✦ 12.14 Perform filter after grouping over a small file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/user_info_reg.csv").import@tc()</code>	/Read file
2	<code>=A1.groups(user_id;sum(reg_time):total_reg)</code>	/Perform sum after grouping by user_id
3	<code>=A2.select(total_reg<1000)</code>	/Filter

Results A3:

user_id	total_reg
41	512
68	130
90	486
...	...

✦ 12.15 Deduplication for a small file



Query text file data to remove duplicates.
Find all unique user IDs based on *user_info_reg.csv*.

Data in the file:

user_id	reg_mon	age	cell_province	id_province	id_city	insertdate	reg_time
483833	2017-04	19	c29	c26	c26241	2018-12-11	56558
156772	2016-05	31	c11	c11	c11159	2018-02-13	81617
...

✦ 12.15 Deduplication for a small file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/user_info_reg.csv").import@tc(user_id)</code>	/Read the specified field
2	<code>=A1.id(user_id)</code>	/Deduplicate user_ids to get unique ones

A2's result

Member
...
928193
928194
928195

✦ 12.16 Count distinct for small file data



Query a text file, removes duplicate data and perform count.
Deduplicate data by Date and ProductID, and then count the number of records.

Data in the file:

ID	PID	DATE	QUANTITY	SID
1211	10075052	2010-01-01	84	10225
2474	10098045	2010-01-01	106	10591
10576	10093980	2010-01-01	53	10720
...

✦ 12.16 Count distinct for small file data



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/PRODUCT_SALE.txt").import@t(Date:date,PID)</code>	/Read specified fields
2	<code>=A1.groups(Date,PID)</code>	/Deduplicate
3	<code>=A2.len()</code>	/Count non-duplicate records

Results of A3:

Value
9849397

✦ 12.17 Perform grouping & count distinct in each group over a small file



Query a text file, group records and remove duplicates in each group and perform count.
Count the number of days with sales records for each product based on *PRODUCT_SALE.txt*.

Data in the file:

ID	PID	DATE	QUANTITY	SID
1211	10075052	2010-01-01	84	10225
2474	10098045	2010-01-01	106	10591
10576	10093980	2010-01-01	53	10720
...

✦ 12.17 Perform grouping & count distinct in each group over a small file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/PRODUCT_SALE.txt").import@t(DATE:date,PID)</code>	/Read specified fields
2	<code>=A1.groups(PID;icount(DATE):days_with_sales)</code>	/Group and deduplicate records and then count the number of days with sales records

A2's result

PID	days_with_sales
10000002	89
10000003	111
10000004	101
...	...

✦ 12.18 Associatively query data over multiple files



Query data from two associated small files.

Find employees who have spouses and where the total of the couple's ages is over 80 based on *Employees.txt* and *EmpRel.txt*.

Data in the files:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
...

Emp1	Emp2	Relationship
21	22	Spouse
10	1	Spouse
...

✦ 12.18 Associatively query data over multiple files



SPL script is as follows:

	A	B
1	<code>=file("E:\\txt\\Employees.txt").import@t().keys(ID)</code>	/Set ID as primary key
2	<code>=file("E:\\txt\\EmpRel.txt").import@t()</code>	
3	<code>=A2.select(Relationship=="Spouse")</code>	/Select records that have a spouse from A2
4	<code>>A3.switch(Emp1,A1;Emp2,A1)</code>	/Replace Emp1 and Emp2 fields with corresponding records in EmpRel table
5	<code>=A3.select(age(Emp1.Birthday)+age(Emp2.Birthday)>80)</code>	/Select records with the couple's total age over 80
6	<code>>A5.run(Emp1=Emp1.Name,Emp2=Emp2.Name)</code>	/Replace records with their Name field values

A5's result after A6 is executed

Emp1	Emp2	Relationship
Ryan	Rebecca	Spouse
Ashley	Samantha	Spouse
Christopher	Rachel	Spouse
...

✦ 12.19 Join small files to query non-associative field



Use foreign key objectification to join two small files and query a non-associative field.

Find the Department with the youngest department manager based on *EMPLOYEE.txt* and *DEPARTMENT.txt*.

Data in the files:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
....

DEPTID	NAME	MANAGER
1	Administration	20
2	Finance	2
....

✦ 12.19 Join small files to query non-associative field



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/EMPLOYEE.txt").import@t()</code>	<code>/Read EMPLOYEE.txt</code>
2	<code>=file("E:/txt/DEPARTMENT.txt").import@t()</code>	<code>/Read DEPARTMENT.txt</code>
3	<code>=A2.join(MANAGER,A1:EID,~:manager)</code>	<code>/Get the corresponding records in employee table through the foreign key of department table</code>
4	<code>=A3.minp(manager.(age(BIRTHDAY))).manager.DEPT</code>	<code>/Find the department with the youngest department manager</code>

A4	Value
	Finance

✦ 12.20 Join small associative files into a wide table



Combine data in two small associative files to generate a wide table.

Add the information *in user_info.csv* to *lending_info.csv* to form a wide table.

Data in the files:

user_id	reg_mon	gender	age	cell_province	id_province	id_city	insertdate
483833	2017-04	M	19	c29	c26	c26241	2018/12/11
156772	2016-05	M	31	c11	c11	c11159	2018/2/13
...

user_id	listing_id	auditing_date	due_date	due_amt
498765	5431438	2019/3/12	2019/4/12	138.5903
34524	5443211	2019/3/15	2019/4/15	208.0805
...

A foreign key join requires that the joining field be unique. In this case, *user_id* in *user_info.csv* must be unique.

✦ 12.20 Join small associative files into a wide table



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/lending_info.csv").import@tc()</code>	<code>/Read lending_info.csv</code>
2	<code>=file("E:/txt/user_info.csv":"utf-8").import@tc()</code>	<code>/Read user_info.csv using the character set "utf-8"</code>
3	<code>=A2.group@1s(user_id)</code>	<code>/Deduplicate user_id by getting the first record of each group and get a unique user_id</code>
4	<code>=A3.fname().m(2:)</code>	<code>/List other user information besides user_id</code>
5	<code>=A1.join(user_id,A3:user_id,{A4.concat@c()})</code>	<code>/Join two tables to form a wide table</code>

A5

user_id	listing_id	auditing_date	due_date	due_amt	reg_mon	gender	age	cell_province	id_province	id_city	insertdate
498765	5431438	2019-03-12	2019-04-12	138.5903	2017-05	M	37	c11	c11	c11245	2019-03-11
34524	5443211	2019-03-15	2019-04-15	208.0805	2015-07	M	26	c25	c25	c25074	2019-03-14
821741	5461707	2019-03-22	2019-04-22	421.2097	2018-03	F	25	c22	c22	c22308	2019-03-21
...

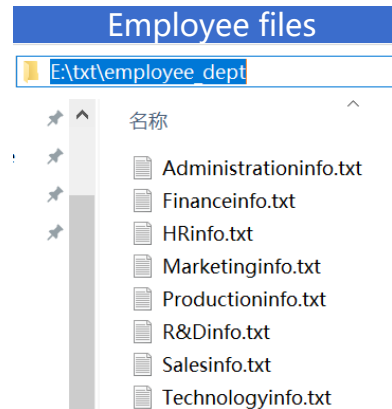
✦ 12.21 Combine data from multiple text files



Combine data from multiple text files.

The employee data of each department is stored in different files under the same directory.

Combine employee data , sort it by *EID*, and save the result as a single file.



✦ 12.21 Combine data from multiple text files



SPL script is as follows:

	A	B
1	<code>=directory@p("E:/txt/employee_dept")</code>	/List files with full pathnames in the directory
2	<code>=A1.(file(~).import@t())</code>	/Read each of the files
3	<code>=A2.conj().sort(EID)</code>	/Combine and sort
4	<code>=file("E:/txt/EMPLOYEE.txt").export@t(A3)</code>	/Write data to a single file

A3

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
...

✦ 12.22 Divide data in a text file into groups and write them to different files



Divide data in a text file into groups and write them to different files.
Write the *EMPLOYEE.txt* file to different files by department.

Data in the file:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
...

✦ 12.22 Divide data in a text file into groups and write them to different files



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/EMPLOYEE.txt").import@t()</code>	<code>/Read EMPLOYEE.txt</code>
2	<code>=A1.group(DEPT)</code>	<code>/Group by department</code>
3	<code>=A2.(file("E:/txt/employee_s/emp_" + ~.DEPT + ".txt").export@t(~))</code>	<code>/Name and export the files</code>

Index	Member
1	[[18,Jonathan,Moore, ...],[20,Alexis,Allen, ...],[26,...
2	[[2,Ashley,Wilson, ...],[13,Daniel,Davis, ...],[23,J...
3	[[4,Emily,Smith, ...],[9,Victoria,Davis, ...]
4	[[8,Megan,Wilson, ...],[17,Hannah,Joh...
5	[[16,Christopher,Hernandez, ...],[19,Samantha,...
6	[[1,Rebecca,Moore, ...],[5,Ashley,Smith, ...],[10,...
7	[[3,Rachel,Johnson, ...],[6,Matthew,Johnson, ...]
8	[[55,Olivia,Anderson, ...],[56,Jacob,Smith, ...],[8...

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	18	Jonathan	Moore	M	Florida	1971-03-07	2000-03-07	Administrat...	7000
2	20	Alexis	Allen	F	Florida	1977-08-07	2007-08-07	Administrat...	16000
3	26	Timothy	Miller	M	Florida	1977-12-24	2007-12-24	Administrat...	5000
4	42	Michael	Jones	M	Pennsylva...	1978-08-20	2008-08-20	Administrat...	12000

File directory	
E:\txt\employee_s	
名称	
emp_Administration.txt	
emp_Finance.txt	
emp_HR.txt	
emp_Marketing.txt	
emp_Production.txt	
emp_R&D.txt	
emp_Sales.txt	
emp_Technology.txt	

✦ 12.23 Write data in a text file to different files according to judgements



Write the data in a text file to different files according to judgements of the specified condition.
Write the *EMPLOYEE_nan.txt* file to different files according to whether there is missing information or not.

Data in the file:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
....	
15	Alexis	Smith	F	New York	1983-07-10	2006-07-10	Sales	8000.0
16	Christopher		M	Florida	1979-06-27	2007-06-27	Production	9000.0
17	Hannah	Johnson	F	Texas		2006-07-19	Marketing	4000.0
....	

✦ 12.23 Write data in a text file to different files according to judgements



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/EMPLOYEE_nan.txt").import@t()</code>	<code>/Import data</code>
2	<code>=[true,false]</code>	<code>/Ensure that the data will be divided into two groups</code>
3	<code>=A1.align@a(A2,~.array().pos(null)>0)</code>	<code>/Divide data into two groups by whether there is missing value or not</code>
4	<code>=A3.(file("E:/txt/employee_N_s/employee_"+["NA","NO_NA"](#)+".txt").export@t(~))</code>	
	<code>/Export to corresponding files respectively</code>	

A3

Index	Member
1	[[16,Christopher,,...],[17,Hannah,Johnson,...],[23,Joseph,,...],...]
2	[[1,Rebecca,Moore,...],[2,Ashley,Wilson,...],[3,Rachel,Johnso...]

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	16	Christopher	(null)	M	Florida	1979-06-27	2007-06-27	Production	9000.0
2	17	Hannah	Johnson	F	Texas	(null)	2006-07-19	Marketing	4000.0
3	23	Joseph	(null)	M	California	1983-08-27	2003-08-27	Finance	6000.0

File directory

E:\txt\employee_N s
名称
employee_NA.txt
employee_NO_NA.txt



Chapter 13

SPL COOKBOOK

Using structured big text file

✦ 13.1 Filter a big file



Filter a big text file to get records meeting the specified condition.
Find scores of students in class 10 in big file *students_scores.txt*.

Text content

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
1	Allen Ashley	98	97	97
1	Allen Brandon	93	76	78
.....				

✦ 13.1 Filter a big file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores.txt").cursor@t()</code>	/Read file by cursor, @t option reads the first line as title
2	<code>=A1.select(CLASS==10)</code>	/Select records of students in class 10, which will be calculated later
3	<code>=A2.fetch()</code>	/Perform A2's calculation and fetch data from cursor,

Results of A3

CLASS	Name	English	Chinese	Math
10	Adams Ashley	89	49	91
10	Adams Kayla	85	74	45
10	Allen Danielle	62	77	88
...

✦ 13.2 Perform aggregate sum over a big text file



Aggregate data in a big text file to get sum.

Calculate the total score of Chinese based on the big text file *students_scores.csv*.

Text content

```
CLASS,NAME,English,Chinese,Math
1,Adams Brooke,63,31,69
1,Adams Hannah,89,85,79
1,Adams Jonathan,88,87,91
...
```

✦ 13.2 Perform aggregate sum over a big text file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores.csv").cursor@tc()</code>	/Read file by cursor; @c means that the specified separator is " , "
2	<code>=A1.total(sum(Chinese))</code>	/Get the sum of Chinese scores

Results of A2

Value
181025

✦ 13.3 Inter-column calculation in a big text file



Perform inter-column calculations in a big text file.

Calculate the total score of each student based on the big text file *students_scores.txt*.

Text content

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...				

✦ 13.3 Inter-column calculation in a big text file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores_.txt").cursor@t(;" ")</code>	/Read file by cursor, the specified separator is " "
2	<code>=A1.derive(English+Chinese+Math:total_score)</code>	/Attach a derive operation to cursor to calculate the total score
3	<code>=A2.fetch@x(100)</code>	/Perform A2's attached calculation, fetch data and close the cursor

A3's result

CLASS	Name	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...

✦ 13.4 Perform comprehensive calculations over a big text file



Perform comprehensive calculations using a big text file.

Calculate the students' average score of Chinese in class 10 and the Chinese average score of students who get a pass for Chinese in class 10 respectively based on big data file *students_scores.txt*.

Text content

```
CLASS|NAME|English|Chinese|Math
1|Adams Brooke|63|31|69
1|Adams Hannah|89|85|79
1|Adams Jonathan|88|87|91
...
```

✦ 13.4 Perform comprehensive calculations over a big text file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores_.txt").cursor@t(CLASS,Chinese;," ")</code>	/Read fields <i>Class</i> and <i>Chinese</i> by cursor
2	<code>=A1.select(CLASS==10)</code>	/Attach the <i>Select</i> calculation to cursor
3	<code>=A2.total(avg(Chinese),avg(if(Chinese>=60,Chinese)))</code>	/Calculate the average, and the average of students who get a pass respectively

A3's result

Member
62.667
78.706

✦ 13.5 Sort a big text file



Sort the structured data in a big text file by the values of a certain field in ascending order.
Sort records by Chinese score in ascending order based on big data file *students_score.txt*.

Some data in the file

Name	Math	Chinese	English
Natalie	84	90	84
Jessica	87	88	78
Brianna	89	90	75
...

✦ 13.5 Sort a big text file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_score.txt").cursor@t()</code>	/Create a cursor
2	<code>=A1.sortx(Chinese)</code>	/Sort in ascending order, and return a cursor
3	<code>=A2.fetch@x(100)</code>	/Get the top 100 records

A3's result

Name	Math	Chinese	English
Hannah	90	76	95
Tyler	87	78	93
Zachary	75	81	85
...

✦ 13.6 Sort a big text file in descending order



Sort the structured data in a big text file in descending order.

Sort records by total score in descending order base on big data file *students_score.txt*.

Some data in the file

Name	Math	Chinese	English
Natalie	84	90	84
Jessica	87	88	78
Brianna	89	90	75
...

✦ 13.6 Sort a big text file in descending order



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_score.txt").cursor@t()</code>	/Create a cursor
2	<code>=A1.sortx(-(Math+English+Chinese))</code>	/Sort by the target values in descending order, and return a cursor
3	<code>=A2.fetch@x(100)</code>	/Fetch data

A3's result

Name	Math	Chinese	English
Emma	88	84	94
Sean	98	86	81
Hannah	90	76	95
...

✦ 13.7 Sort a big text file by multiple fields in specified order



Sort data in a big text file by multiple fields in the specified order.
Sort records by class in ascending order, and by total score in descending order
based on big data file *students_scores.txt*.

Some data in the file

Name	Math	Chinese	English
Natalie	84	90	84
Jessica	87	88	78
Brianna	89	90	75
...

✦ 13.7 Sort a big text file by multiple fields in specified order



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/students_scores.txt").cursor@t()</code>	/Create a cursor
2	<code>=A1.sortx(CLASS,-(English+Chinese+Math))</code>	/Sort according to requirements, and return a cursor
3	<code>=A2.fetch@x(100)</code>	/Fetch data

A3's result

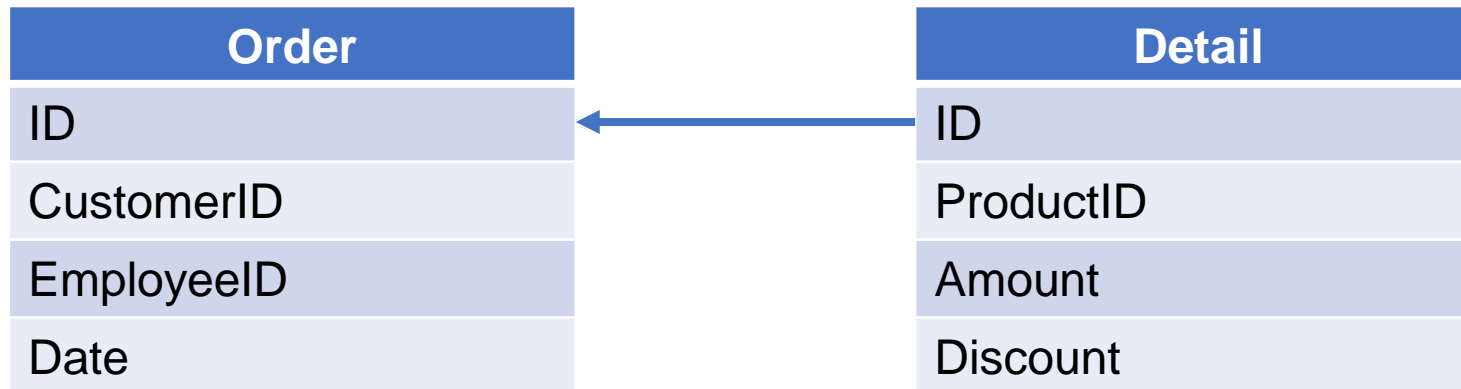
	CLASS	Name	English	Chinese	Math
	1	Allen Ashley	98	97	97
	1	Lewis Antony	93	92	94
	1	Adams Jonathan	88	87	91

✦ 13.8 Find records in a big data table that match data in another big data table



Find records in a big data table that match the filtered data in another big data table.

Based on two big data tables, *order* and *detail*, query the number of orders with actual sales amount exceeding 1000 for each salesman, and sort the final records in descending order.



✦ 13.8 Find records in a big data table that match data in another big data table



The SQL query:

```
select
    EmployeeID, count(1) as OrderCount
from
    Order
where
    ID in ( select ID from Detail where Amount*(1-Discount) > 1000 )
group by EmployeeID
order by OrderCount desc
```

✦ 13.8 Find records in a big data table that match data in another big data table



The joinx() function is used here perform order-based merge. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.cursor("select * from Order order by ID")	/Create a cursor for <i>Order</i> table
3	=A1.cursor("select * from Detail order by ID")	/Create a cursor for <i>Detail</i> table
4	=A3.select(Amount*(1-Discount)>1000)	/Select records with actual sales amount over 1000
5	=A4.group@1(ID)	/Group <i>Detail</i> by order ID, and only get one record for each group
6	=joinx(A2:Order,ID;A5:Detail,ID)	/Use joinx function to merge the cursors of <i>Order</i> table and <i>Detail</i> table in order
7	=A6.groups(Order.EmployeeID:EmployeeID;count(~):OrderCount).sort@z(OrderCount)	/Group by EmployeeID and count the orders of each employee, with records sorted by OrderCount in descending order

A7	EmployeeID	OrderCount
	4	43
	3	42

✦ 13.9 Perform grouping & aggregation over a big file, with small result set



Perform grouping & aggregation over a big text file.

Count the total number of logins in each province based on big data file *user_info_reg.csv*.

Some data in the file

user_id	reg_mon	age	cell_province	id_province	id_city	insertdate	reg_time
483833	2017-04	19	c29	c26	c26241	2018-12-11	56558
156772	2016-05	31	c11	c11	c11159	2018-02-13	81617
...

✦ 13.9 Perform grouping & aggregation for a large file, with small result set



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/user_info_reg.csv").cursor@tc()</code>	/Create a cursor
2	<code>=A1.groups(id_province;count(~):cnt)</code>	/Perform count after grouping

Results of A2

id_province	cnt
c01	27202
c02	61735
c03	14433
...	...

✦ 13.10 Perform grouping & aggregation over a big file, with large result set



Perform grouping & aggregation over a big text file. External storage is used for calculation because the result set is large.

Calculate the total login duration of each user based on big data file *user_info_reg.csv*.

Some data in the file

user_id	reg_mon	age	cell_province	id_province	id_city	insertdate	reg_time
483833	2017-04	19	c29	c26	c26241	2018-12-11	56558
156772	2016-05	31	c11	c11	c11159	2018-02-13	81617
...

✦ 13.10 Perform grouping & aggregation for a large file, with large result set



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/user_info_reg.csv").cursor@tc()</code>	/Create a cursor
2	<code>=A1.groupx(user_id;sum(reg_time):total_reg)</code>	/Perform sum after grouping, and return a cursor
3	<code>=A2.fetch@x(1000)</code>	

Results of A3

user_id	total_reg
1	2345
2	74990
3	53724
...	...

✦ 13.11 Filter after grouping over a big file



Perform filtering after grouping over a big text file.

Find the users whose total login duration is less than 1000 minutes based on big data file *user_info_reg.csv*.

Some data in the file

user_id	reg_mon	age	cell_province	id_province	id_city	insertdate	reg_time
483833	2017-04	19	c29	c26	c26241	2018-12-11	56558
156772	2016-05	31	c11	c11	c11159	2018-02-13	81617
...

✦ 13.11 Filter after grouping over a big file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/user_info_reg.csv").cursor@tc()</code>	/Create a cursor
2	<code>=A1.groupx(user_id;sum(reg_time):total_reg)</code>	/Perform sum after grouping and return a cursor
3	<code>=A2.select(total_reg<1000).fetch()</code>	/Perform filtering and fetch data

A3's result

user_id	total_reg
41	512
68	130
90	486
...	...

✦ 13.12 Deduplication of big text file



Query a big text file and remove duplicate data.

Find all unique user IDs based on big data file *user_info_reg.csv*.

Some data in the file

user_id	reg_mon	age	cell_province	id_province	id_city	insertdate	reg_time
483833	2017-04	19	c29	c26	c26241	2018-12-11	56558
156772	2016-05	31	c11	c11	c11159	2018-02-13	81617
...

✦ 13.12 Deduplication of big text file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/user_info_reg.csv").cursor@tc()</code>	/Create a cursor
2	<code>=A1.id(user_id)</code>	/Deduplicate to find unique user_ids

A2's result

Member
...
928193
928194
928195

✦ 13.13 Count distinct over a big text file



Query a big text file, remove duplicate data and perform count.

Based on the big data file *PRODUCT_SALE.txt*, deduplicate data by date and product, and then count the number of records.

Some data in the file

ID	PID	DATE	QUANTITY	SID
1211	10075052	2010-01-01	84	10225
2474	10098045	2010-01-01	106	10591
10576	10093980	2010-01-01	53	10720
...

✦ 13.13 Count distinct over a big text file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/PRODUCT_SALE.txt").cursor@t(Date,PID)</code>	/Read the specified fields
2	<code>=A1.groupx(date(Date),PID)</code>	/Deduplicate
3	<code>=A2.skip()</code>	/Calculate the number of non-repeated records

A3's result

Value
9849397

✦ 13.14 Group & count distinct in each group over a big text file



Query a big data text file, group data and then remove duplicates in each group and perform count.
Count the number of days with sales records for each product based on big data file *PRODUCT_SALE.txt*.

Some data in the file

ID	PID	DATE	QUANTITY	SID
1211	10075052	2010-01-01	84	10225
2474	10098045	2010-01-01	106	10591
10576	10093980	2010-01-01	53	10720
...

✦ 13.14 Group & count distinct in each group over a big text file



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/PRODUCT_SALE.txt").cursor@t(Date,PID)</code>	/Create a cursor
2	<code>=A1.groupx(date(Date),PID)</code>	/Deduplicate
3	<code>=A2.groups(PID;count(1):days_with_sales)</code>	/Group records and count the number of days with sales records

A3's result

PID	days_with_sales
10000002	89
10000003	111
10000004	101
...	...

✦ 13.15 Group a big file by values of a certain field, and query record containing the max value of another field in each group



Group records in a big text file, find one record in each group by certain conditions, then merge them to return. Query the records with the highest monthly amount based on big data file *Sales*.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...



13.15 Group a big file by values of a certain field, and query record containing the max value of another field in each group



SPL script is as follows, where `cs.group(x, ...)` function groups records by comparing adjacent values and returns the grouped original cursor:

	A	B
1	<code>=connect("db").query("select * from Sales order by OrderDate")</code>	<code>/ReadSales</code> table in the database and sort it by <code>OrderDate</code>
2	<code>=A1.group(month(OrderDate))</code>	<code>/cs.group()</code> function groups records by comparing the adjacent months
3	<code>=A2.(~.maxp(Amount))</code>	<code>/Select</code> the record with the highest monthly sales for each month
4	<code>=A3.conj()</code>	<code>/Return</code> the concatenation of records
5	<code>=A4.fetch()</code>	<code>/Fetch</code> data from cursor; the result set is small

A5	OrderID	Customer	SellerId	OrderDate	Amount
	10267	FRANK	4	2013/07/29	4031.0
	10286	QUICK	8	2013/08/21	3016.0

✦ 13.16 Combine & calculate data in multiple big data files



Combine & calculate records in multiple big text files.

Based on two big files *s2014* and *s2015* that contain sales data, query the customers whose total sales rank in top three in the past two years.

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 13.16 Combine & calculate data in multiple big data files



SPL script is as follows, where CS.conjx() function concatenates multiple cursors vertically, which is equivalent to combining records in the cursors:

	A	B
1	=connect("db")	/Connect to database
2	=A1. cursor("select * from S2014")	/Get the cursor of table S2014
3	=A1. cursor("select * from S2015")	/Get the cursor of table S2015
4	=A2,A3].conjx()	/CS.conjx() function concatenates multiple cursors
5	=A4.groups(Customer; sum(Amount):Amount)	/Group and aggregate the combined cursor, and calculate the total sales of each customer
6	=A5.top(-3;Amount)	/Select customers with top3 total sales in two years

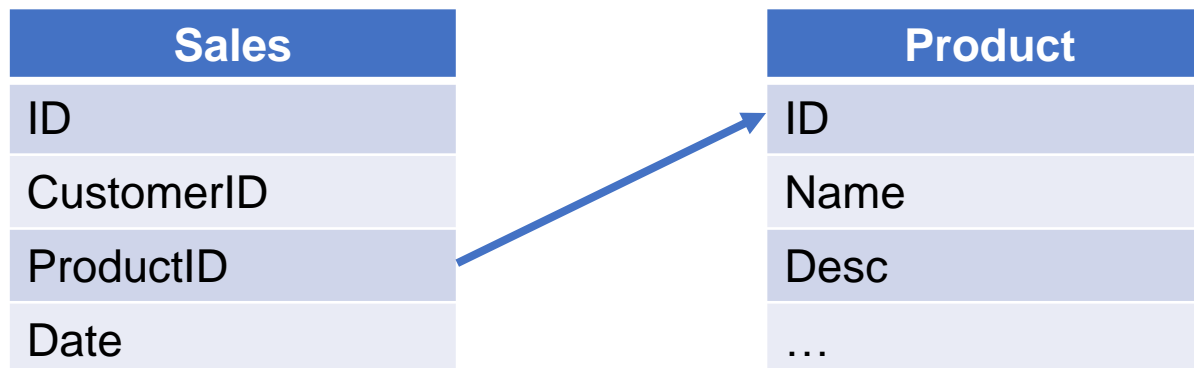
A6	Customer	Amount
	SAVEA	177478.89
	QUICK	102764.99
	ERNSH	94066.28

✦ 13.17 The join filter over a large file and a small file



Associatively query data over a large file and a small file.

Product information and sales information is stored in two text files respectively. Calculate the total sales amount of products with each order quantity being not more than 10. The *Sales* table has a large amount of data and cannot be wholly read into memory.



✦ 13.17 The join filter over a large file and a small file



SPL script is as follows :

	A	B
	1 =file("E:/txt/Products.txt").import@t()	/Read <i>Products</i> (in-memory table) and create an index
	2 =file("E:/txt/Sales.txt").cursor@t()	/Read <i>Sales</i> table as a single cursor or a multicursor
	3 =A2.select(quantity<=10)	/Filter the cursor
Method 1	4 =A3.switch(productid,A1:ID)	/The switch function replaces foreign key of the cursor with the corresponding records in the other table
	5 =A4.groups(;sum(quantity*productid.Price):total)	/Aggregate to get sum
Method 2	4 =A3.join(productid,A1:ID,~:products)	/The join function attach records corresponding to foreign key to the cursor
	5 =A4.groups(;sum(quantity*products.Price):total)	/Aggregate to get sum
Method 3	4 =A3.join(productid,A1:ID,Price)	/Use join to add Price field
	5 =A4.groups(;sum(quantity*Price):total)	/Aggregate to get sum

Results of A5

total
142740.180

✦ 13.18 Join a large file and a small file into a wide table to query



Join a big data file and a small file into a wide table, and then perform query.
Add the information *in user_info.csv* to *lending_info.csv* to form a wide table.

Some data in the files:

user_id	reg_mon	gender	age	cell_province	id_province	id_city	insertdate
483833	2017-04	M	19	c29	c26	c26241	2018/12/11
156772	2016-05	M	31	c11	c11	c11159	2018/2/13
...

user_id	listing_id	auditing_date	due_date	due_amt
498765	5431438	2019/3/12	2019/4/12	138.5903
34524	5443211	2019/3/15	2019/4/15	208.0805
...

✦ 13.18 Join a large file and a small file into a wide table to query



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/lending_info.csv").cursor@tc()</code>	<code>/Create a cursor</code>
2	<code>=file("E:/txt/user_info.csv":"utf-8").import@tc()</code>	<code>/Read user_info.csv using the character set"utf-8"</code>
3	<code>=A2.group@1s(user_id)</code>	<code>/Deduplicate user_id by getting the first record of each group to get unique user_ids</code>
4	<code>=A3.fname().m(2:)</code>	<code>/List other information of other users besides user_id</code>
5	<code>=A1.join(user_id,A3:user_id,\${A4.concat@c()})</code>	<code>/Join the smaller tables by cursor to form a wide table, and return a cursor</code>
6	<code>=A5.fetch@x(100)</code>	<code>/Fetch 100 row and close the cursor</code>

A6

user_id	listing_id	auditing_date	due_date	due_amt	reg_mon	gender	age	cell_province	id_province	id_city	insertdate
498765	5431438	2019-03-12	2019-04-12	138.5903	2017-05	M	37	c11	c11	c11245	2019-03-11
34524	5443211	2019-03-15	2019-04-15	208.0805	2015-07	M	26	c25	c25	c25074	2019-03-14
821741	5461707	2019-03-22	2019-04-22	421.2097	2018-03	F	25	c22	c22	c22308	2019-03-21
...

✦ 13.19 Merge-join two big files



Perform merge join over two big data files to query.

The orders table and order details table are stored in two big data files. Calculate the total sales amount of each customer.

Some data in *Orders* table

orderid	clientid	date
10012	100658	2019-02-13
10023	103478	2019-01-12
10040	108013	2019-01-04
...

Some data in *Orderdetails* table

orderid	no	productid	price
10012	1	3018	428.5
10012	2	3019	349.2
10023	1	3019	349.2
...

✦ 13.19 Merge-join two big files



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/Orders.txt").cursor@t().sortx(orderid)</code>	<code>/sortx is not needed if data is already ordered by orderid</code>
2	<code>=file("E:/txt/OrderDetails.txt").cursor@t().sortx(orderid)</code>	
3	<code>=joinx(A1:order,orderid;A2: detail,orderid)</code>	<code>/Use joinx to join two cursors</code>
4	<code>=A3.groups(order.clientid:clientid;sum(detail.price):amount)</code>	<code>/Calculate the total sales amount of each customer</code>

A4's result

clientid	amount
100008	12350.0
100011	53400.0
100015	13790.0
...	...

✦ 13.20 Set operations of multiple big text files



Set operations of multiple big text files: external storage is needed because of the large amount of data. User login information is stored in different files by month. Query login information of a large number of users according to the requirements.

✦ 13.20 Set operations of multiple big text files



SPL script is as follows:

	A	B
1	<code>=file("E:/txt/user_login_info_1.txt").cursor@t().sortx(userid).group@1(userid)</code>	/Users' first login information in Jan, Feb and Mar. sortx is not needed if data is already ordered by userid
2	<code>=file("E:/txt/user_login_info_2.txt").cursor@t().sortx(userid).group@1(userid)</code>	
3	<code>=file("E:/txt/user_login_info_3.txt").cursor@t().sortx(userid).group@1(userid)</code>	
4	<code>=[A1,A2,A3].mergex(userid).fetch()</code>	/Merge users' first login information of each month in order by userid
4	<code>=[A1,A2,A3].mergex@u(userid).fetch()</code>	/Get the union, i.e. users who log in at least once in 3 months
4	<code>=[A1,A2,A3].mergex@i(userid).fetch()</code>	/Get the intersection, i.e. users who log in every month for 3 months
4	<code>=[A1,A2,A3].mergex@d(userid).fetch()</code>	/Get the difference, i.e., users who log in in January but not in February and March

userid	login
600001	2019-01-25 02:49:43
600001	2019-02-11 13:16:46
600001	2019-03-06 17:27:49
...	...

userid	login
600001	2019-01-25 02:49:43
600002	2019-01-20 03:00:28
600003	2019-01-13 14:34:20
...	...

userid	login
600001	2019-01-25 02:49:43
600002	2019-01-20 03:00:28
600003	2019-01-13 14:34:20
...	...

userid	login
601293	2019-01-10 08:04:36
601999	2019-01-11 16:49:25
605227	2019-01-18 15:24:26
...	...

The cursor can only traverse one way when data is fetched from it, so only one of the four operations in A4 can be executed at a time.

✦ 13.21 Divide a big text file into groups and write them to different files



Divide data in a big data text file into groups and write them to different files. External storage is needed because of the large amount of data.

Write the big data file *EMPLOYEE.txt* to different files by department.

Some data in the file

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
...

✦ 13.21 Divide a big text file into groups and write them to different files



SPL script is as follows:

	A	B
1	=file("E:/txt/EMPLOYEE.txt").cursor@t()	
2	for A1,10000	=A2.group(DEPT)
3		=B2.(file("E:/txt/EMPLOYEE/EMP_"+~.DEPT+".txt").export@a@at(~))
4	/Read file by cursor and fetch data in loop. The data retrieved each time is processed as a small file. @a is used to append data during the export.	

Results of A2, B2 in the first loop

File directory

E:\txt\EMPLOYEE	
名称	
EMP_Administration.txt	
EMP_Finance.txt	
EMP_HR.txt	
EMP_Marketing.txt	
EMP_Production.txt	
EMP_R&D.txt	
EMP_Sales.txt	
EMP_Technology.txt	

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	1	Rebecca	Moore	F	California	1974-11-20	2005-03-11	R&D	7000
2	2	Ashley	Wilson	F	New York	1980-07-19	2008-03-16	Finance	11000
3	3	Rachel	Johnson	F	New Mexico	1970-12-17	2010-12-01	Sales	9000
4	4	Emily	Smith	F	Texas	1985-03-07	2006-08-15	HR	7000
5	5	Ashley	Smith	F	Texas	1975-05-13	2004-07-30	R&D	16000

Index	Member
1	[[18,Jonathan,Moore, ...],[20,Alexis,Allen, ...],[26,Timothy,Miller, ...], ...]
2	[[2,Ashley,Wilson, ...],[13,Daniel,Davis, ...],[23,Joseph,Turner, ...], ...]
3	[[4,Emily,Smith, ...],[9,Victoria,Davis, ...],[51,Madison,Willia, ...], ...]
4	[[8,Megan,Wilson, ...],[17,Hannah,Johnson, ...],[21,Jacob, ...], ...]
5	[[16,Christopher,Hernandez, ...],[19,Samantha,Williams, ...], ...]

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	18	Jonathan	Moore	M	Florida	1971-03-07	2000-03-07	Administrat...	7000
2	20	Alexis	Allen	F	Florida	1977-08-07	2007-08-07	Administrat...	16000

✦ 13.22 Write data in a large text file to different files according to judgements



Write data in a big data text file to different files according to the judgement of the specified condition. External storage is needed because of the large amount of data.

Write the big data file *EMPLOYEE_nan.txt* to different files according to whether there is missing information or not.

Some data in the file

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
....	
15	Alexis	Smith	F	New York	1983-07-10	2006-07-10	Sales	8000.0
16	Christopher		M	Florida	1979-06-27	2007-06-27	Production	9000.0
17	Hannah	Johnson	F	Texas		2006-07-19	Marketing	4000.0
....	

✦ 13.22 Write data in a large text file to different files according to judgements



SPL script is as follows:

	A	B
1	=file("E:/txt/EMPLOYEE_nan.txt").cursor@t()	
2	=[true,false]	/Ensure that ata is divided into two groups each time
3	for A1,10000	=A3.align@a(A2,~.array().pos(null)>0)
4		=B2.(file("E:/txt/EMPLOYEE_N/EMPLOYEE_"+"["NA","NO_NA"](#)+".txt").export@at(~))
5	/Read file by cursor and fetch data in loop. The data retrieved each time is processed as a small file. @a is used to append data during the export.	

Results of A3, B3 of the first loop

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	1	Rebecca	Moore	F	California	1974-11-20	2005-03-11	R&D	7000.0
2	2	Ashley	Wilson	F	New York	1980-07-19	2008-03-16	Finance	11000.0
3	3	Rachel	Johnson	F	New Mexico	1970-12-17	2010-12-01	Sales	9000.0
4	4	Emily	Smith	F	Texas	1985-03-07	2006-08-15	HR	7000.0
5	5	Ashley	Smith	F	Texas	1975-05-13	2004-07-30	R&D	16000.0

Index	Member
1	[[16,Christopher,,...],[17,Hannah,Johnson,...],[23,Joseph,...]]
2	[[1,Rebecca,Moore,...],[2,Ashley,Wilson,...],[3,Rachel,Johnson,...]]

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	16	Christopher	(null)	M	Florida	1979-06-27	2007-06-27	Production	9000.0
2	17	Hannah	Johnson	F	Texas	(null)	2006-07-19	Marketing	4000.0
3	23	Joseph	(null)	M	California	1983-08-27	2003-08-27	Finance	6000.0
4	27	Alexis	Jones	F	California	1983-12-27	(null)	Marketing	10000.0

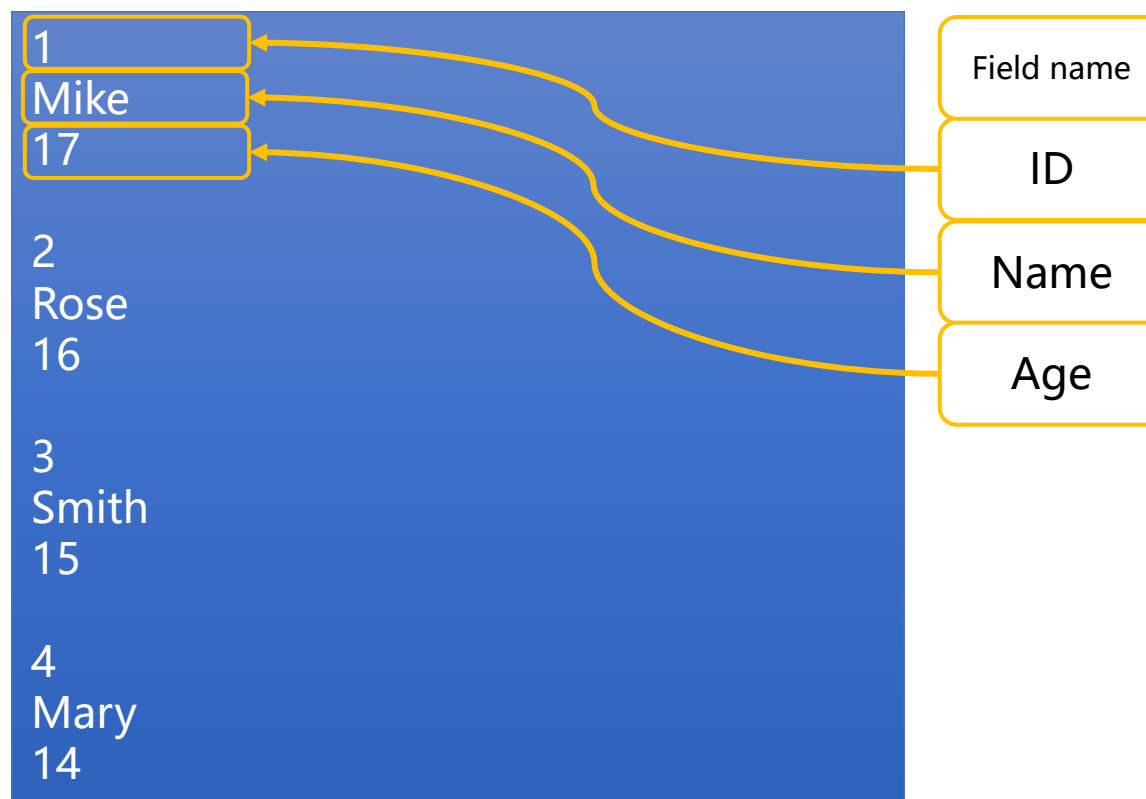
File directory	
E:\txt\EMPLOYEE_N	
名称	^
	EMPLOYEE_NA.txt
	EMPLOYEE_NO_NA.txt

✦ 13.23 Organize a fixed-structure big text file into structured data



Organize a multiline big text file with fixed structure into structured data.

In the big data file student.txt, every three line is a record, and there is a blank line between adjacent records. Read the text and organize it into structured data.



✦ 13.23 Organize a fixed-structure big text file into structured data



By default, the cursor stores data as a structured table sequence. But since this text file has only one column, we use @i option to convert the one-column table to a sequence, which is convenient for subsequent calculation.

Note the difference between the cursor data and the row read in the last section. The cursor always stores data in a certain structure, and automatically resolves data to the proper data type, so the empty row is resolved as null.

The data is processed in loop and blocks. It should be noted that the number of blocks fetched each time must be a multiple of 3, because the current table structure is 3 columns; otherwise the data filling will be misplaced.

After the structured data is calculated, clean it up to prevent memory overflow.

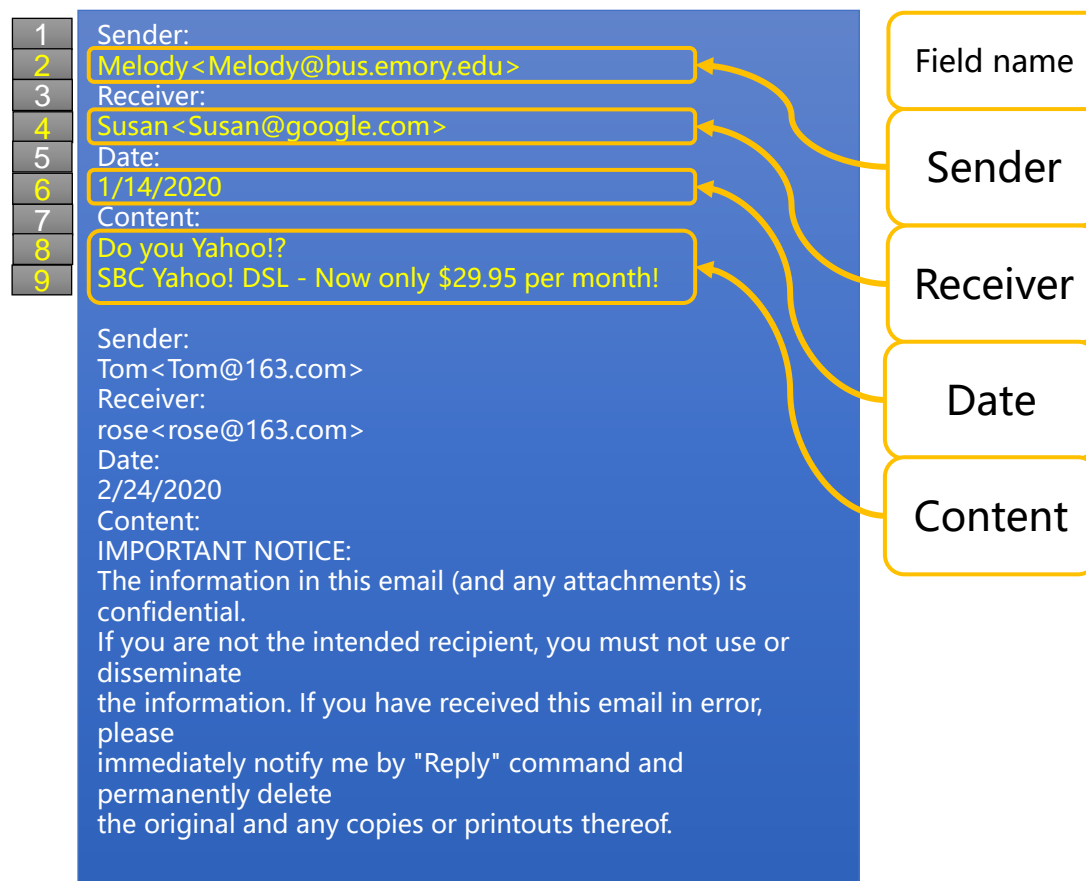
	A	B
1	=file("D:\\student.txt")	/Open the file
2	=A1.cursor@i()	/Create file cursor
3	=A2.select(~!=null)	/Remove blank lines between records
4	=create(ID,Name,Age)	/Create table structure
5	for A3,3000	>A4.reset()
6		=A4.record(A5)
7		=file("D:\\struct_student.txt").export@at(B6)

✦ 13.24 Organize a big file with indefinite-line structure into structured data



Organize a big text file with indefinite-line structure into structured data.

Sort out the data in the big file *mail.txt*, including the sender, receiver, date, mail content, etc.. The contents of mails may contain different numbers of lines.



✦ 13.24 Organize a big file with indefinite-line structure into structured data



SPL script is as follows:

	A	B
1	<code>=file("D:\\mail.txt")</code>	/Open the file
2	<code>=A1.cursor@i().select(~!=null)</code>	/Create a cursor and attach the operation of removing empty lines to it
3	<code>=A2.group@i(~=="Sender:")</code>	/Group by condition
4	<code>=A3.new(~(2):Sender,~(4):Receiver,~(6):Date,~to(8).concat():Content)</code>	/Extract record values, merge the contents, create new structure table
5	<code>=A4.fetch@x(100)</code>	/Fetch 100 records from cursor

Use @i option to read data as a sequence, and use *null* to identify empty lines.

Fetch data from a structured table. Data needs to be fetched in blocks to prevent memory overflow.

`~to(8)` means starting from line 8 of the current group; omitting parameters after the comma means getting all the following lines.
`.concat()` concatenates members of a sequence into a string.

✦ 13.25 Find the lines containing keyword in all big text files in the specified directory



Traverse all big text files in the specified directory, read data from each file in batches, and find the lines containing the keyword.

✦ 13.25 Find the lines containing keyword in all big text files in the specified directory



SPL script is as follows:

Define two entry parameters: path is the root directory, and key is the keyword to be searched.

@s option processes data by rows; @i option converts the returned single-field table sequence to sequence; run() traverses files to search the contents of each file.

Fetch data from B2's cursor in loop, with 1000 rows fetched each time, until all data is fetched. The run function defined earlier will be executed automatically during the fetching process.

	A	B	C
1	=directory@ps(path+ "/*.txt")	/List all text files in the directory(including subdirectories)	
2	for A1	=file(A2).cursor@is().run(if(pos(~,key),output(A2/"No"/#/"Row: "/~)))	/Create file cursor
3		for B2,1000	/Fetch data in blocks
4			

✦ 13.26 Replace specified text in all text files under the specified directory



Traverse all text files under the directory, read data from each file in batches, replace the specified text and output data in the updated files to new files.

✦ 13.26 Replace specified text in all text files under the specified directory



SPL script is as follows:

Define three entry parameters: path is the root directory, source is the keyword to be searched, and target is the word to replace with.

When doing the replacement in cursor, because the source file reading and updated file writing are carried out at the same time, we need a new file to which the updated data is written.

Read data in loop and append data to B4's file.

	A	B	C
1	=directory@ps(pat h+"/*.txt")	/List all text files in the directory(including subdirectories)	
2	for A1	=file(A2).cursor@is()	/Loop through all files in the directory and create cursor
3		=B2.run(~=replace(~,source,target))	/Define replacement calculation for cursor
4		=file(filename@d(A2)+"\\"+filename@n(A2)+"_2."+filename@e(A2))	/Define a new output file in the path of the source file
5		=movefile(B4)	/Delete a new file with the same name to prevent wrong appending
6		for B3,1000	=B4.write@a(B6)

✦ 13.27 Count the frequencies of each word in a big text file



Count the number of each English word's appearances.

✦ 13.27 Count the frequencies of each word in a big text file



SPL script is as follows:

Define an entry parameter: filepath is the name of the target file.

@is option reads data as a sequence.

	A	B
1	<code>=file(filepath),cursor@is()</code>	/Create file cursor
2	<code>=A1.run(~=~.words()).conj()</code>	/Define delayed calculation. Each row of data is split into a sequence of words and finally all sequences is merged into a large sequence.
3	<code>=A2.groups(~:Word;count(~):Count)</code>	/Perform count over the cursor

✦ 13.28 Count the frequencies of each letter in a big text file



Count the number of appearances of each letter in a big text file.

✦ 13.28 Count the frequencies of each letter in a big text file



SPL script is as follows:

Define an entry parameter: filepath is the name of the target file.

@is option reads data as a sequence.

	A	B
1	<code>=file(filepath).cursor@is()</code>	/Create file cursor
2	<code>=A1.run(~::~split()).conj()</code>	/Define delayed calculation. Each row of data is split into a sequence of words and finally all sequences is merged into a large sequence.
3	<code>=A2.groups(~:Char;count(~):Count)</code>	/Perform count over the cursor

✦ 13.29 Remove duplicate lines from a big text file



Remove duplicate lines from a big text file.

```
https://123.sogou.com/  
https://www.sogou.com/  
https://stackoverflow.com/  
https://123.sogou.com/  
http://www.raqsoft.com.cn/  
https://www.baidu.com/  
https://www.sogou.com/  
https://123.sogou.com/  
https://stackoverflow.com/  
http://www.raqsoft.com.cn/
```


✦ 13.29 Remove duplicate lines from a big text file



SPL script is as follows:

	A	B
1	d:/urls.txt	/Specify the file path
2	=file(A1).cursor@s()	/Open file and create a cursor to read the text
3	=A2.groupx(_1)	/Group by default field name _1
4	=file(filename@d(A1)+"\\"+filename@n(A1)+"_2."+filename@e(A1))	/Construct output file under the same path
5	for A3,1000	=A4.export@a(A5)

@a option means to export by appending.

@s option indicates to return a cursor with only one column, and return a table sequence with default column name when data is fetched.

✦ 13.30 Remove repeated paragraphs from a big text file



Delete repeated paragraphs from a big text file.

✦ 13.30 Remove repeated paragraphs from a big text file



SPL script is as follows:

By removing duplicate by paragraph, the content after the deduplication should not be disrupted. Add row number to each row to restore the original order after grouping.

Note that the sequence number expression is `seq()`, which is different from the symbol `#` used for a table sequence. `#` represents the sequence number of a member in a sequence. When fetching data from a cursor by blocks, block fetched in each round `#` starts from 1. In this case we use `seq()` function to get the correct consecutive sequence number.

When obtaining the unique row of each group:
1: Use `group@1()` for a sequence. `@1` option must be present and the parameter can be absent.
2: Use `group@1(_1)` for a table sequence. Both `@1` option and a field parameter must be present.
3: Use `groupx(_1;min(Row):Row)` for a table sequence's cursor. The option is absent, the grouping field is present, and an aggregate function performed over output rows is present.

	A	B
1	<code>d:/novel.txt</code>	<code>/Specify the file path</code>
2	<code>=file(A1).cursor@s()</code>	<code>/Create a cursor to read text</code>
3	<code>=A2.derive(seq():Row)</code>	<code>/Add a column to calculate row number</code>
4	<code>=A3.groupx(_1;min(Row):Row)</code>	<code>/Group by filed _1, keep the row with the minimum number among duplicate rows</code>
5	<code>=A4.sortx(Row)</code>	<code>/Sort by row number</code>
6	<code>=file(filename@d(A1)+"\\ "+filename@n(A1)+" - 2." + filename@e(A1))</code>	<code>/Construct output file under the same path</code>
7	<code>for A5,1000</code>	<code>=A6.export@a(A7,_1)</code>



Chapter 14

SPL
COOKBOOK

Querying text data directly with SQL

✦ 14.1 Filter



Use SQL to find records that meet a certain condition from a text file.
Get the scores of students in class 10.

Text content

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...				

✦ 14.1 Filter



SPL script is as follows:

	A	B
1	\$select * from E:/txt/Students_scores.txt where CLASS=10	/Filter

A1's result

CLASS	Name	English	Chinese	Math
10	Adams Ashley	89	49	91
10	Adams Kayla	85	74	45
10	Allen Danielle	62	77	88
...

✦ 14.2 Aggregate



Use SQL to aggregate data in a text file.

Calculate the average Chinese score of all students.

Text content	CLASS	NAME	English	Chinese	Math
	1	Adams Brooke	63	31	69
	1	Adams Hannah	89	85	79
	1	Adams Jonathan	88	87	91
	...				

✦ 14.2 Aggregate



SPL script is as follows:

	A	B
1	<code>\$select avg(Chinese) from E:/txt/Students_scores.txt</code>	/Aggregate to get average

A1's result

_1
62.16517857142857

✦ 14.3 Inter-column calculation



Use SQL to do inter-column calculation in a text file.
Calculate the total score of each student.

Text content

CLASS	NAME	English	Chinese	Math	
1	Adams Brooke	63	31	69	
1	Adams Hannah	89	85	79	
1	Adams Jonathan	88	87	91	
...					

✦ 14.3 Inter-column calculation



SPL script is as follows:

	A	B
1	\$select *,English+Chinese+Math as total_score from E:/txt/students_scores.txt	/Add a computed column

A1's result

CLASS	Name	English	Chinese	Math	total_score
1	Adams Brooke	63	31	69	163
1	Adams Hannah	89	85	79	253
1	Adams Jonathan	88	87	91	266
...

✦ 14.4 CASE statement



Case statement can be used in SQL to do calculations with complex conditions.
Calculate whether each student has passed the English test or not.

Text content

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...				

✦ 14.4 CASE statement



SPL script is as follows:

	A	B
1	<pre>\$select *, case when English>=60 then 'Pass' else 'Fail' end as English_evaluation from E:/txt/students_scores.txt</pre>	/Judge whether the English score is 'Pass' or not

A1's result

CLASS	Name	English	Chinese	Math	English_evaluation
1	Adams Brooke	63	31	69	Pass
1	Adams Hannah	89	85	79	Pass
1	Adams Jonathan	88	87	91	Pass
...

✦ 14.5 Sort



Use SQL to sort a text file in ascending/descending order.

Sort records by class in ascending order and by total score in descending order based on *students_scores.txt*.

Text content	CLASS	NAME	English	Chinese	Math
	1	Adams Brooke	63	31	69
	1	Adams Hannah	89	85	79
	1	Adams Jonathan	88	87	91
	...				

✦ 14.5 Sort



SPL script is as follows:

	A	B
1	<pre>\$select * from E:/txt/students_scores.txt order by CLASS,English+Chinese+Math desc</pre>	/Sort by the specified field or an expression

A1's result

CLASS	Name	English	Chinese	Math
1	Allen Ashley	98	97	97
1	Lewis Antony	93	92	94
1	Adams Jonathan	88	87	91
...



Use SQL to get Top-N records in a text file.

Find scores of students whose English scores rank in Top3.

Text content

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...				

✦ 14.6 TOP-N



SPL script is as follows:

	A	B
1	<pre>\$select top 3 * from E:/txt/students_scores.txt order by English desc</pre>	<pre>/top n</pre>

A1's result

CLASS	Name	English	Chinese	Math
2	Davis Tyler	99	62	56
1	Jackson Destiny	99	61	84
5	Williams Andrew	99	66	65

✦ 14.7 Group & Aggregate



Use SQL to group and aggregate data in a text file.
Query the average English score of each class.

Text content

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...				

✦ 14.7 Group & Aggregate



SPL script is as follows:

	A	B
1	<pre>\$select CLASS,avg(English) as avg_En from E:/txt/students_scores.txt group by CLASS</pre>	/Group and aggregate

A1's result

CLASS	avg_En
1	74.43103448275862
2	77.34375
3	72.72857142857143
...	...

✦ 14.8 Filter after grouping



Use SQL to group and aggregate data in a text file and then filter it.
Find the classes whose average English score is less than 70.

Text content

CLASS	NAME	English	Chinese	Math
1	Adams Brooke	63	31	69
1	Adams Hannah	89	85	79
1	Adams Jonathan	88	87	91
...				

✦ 14.8 Filter after grouping



SPL script is as follows:

	A	B
1	<pre>\$select CLASS,avg(English) as avg_En from E:/txt/students_scores.txt group by CLASS having avg(English)<70</pre>	/Filter after grouping

A1's result

CLASS	avg_En
4	69.6046511627907
7	69.86

✦ 14.9 Select distinct



Use SQL to query distinct data in a text file.

Query the class IDs.

Text content	CLASS	NAME	English	Chinese	Math	
	1	Adams Brooke	63	31	69	
	1	Adams Hannah	89	85	79	
	1	Adams Jonathan	88	87	91	
	...					

✦ 14.9 Select distinct



SPL script is as follows:

	A	B
1	<code>\$select distinct CLASS from E:/txt/students_scores.txt</code>	<code>/Use distinct to remove duplicates</code>

A1's result

CLASS
1
2
3
4
5
6
7
8

✦ 14.10 Count distinct



Use SQL to deduplicate and count data in a text file.

Count the number of distinct products in *PRODUCT_SALE.txt*.

Data in the file

ID	PID	DATE	QUANTITY	SID	
1211	10075052	2010-01-01		84	10225
2474	10098045	2010-01-01		106	10591
10576	10093980	2010-01-01		53	10720
...					

✦ 14.10 Count distinct



SPL script is as follows:

	A	B
1	<pre>\$select count(distinct PID) from E:/txt/PRODUCT_SALE.txt</pre>	/count(distinct), which deduplicate data and count distinct values
2	<pre>\$select count(*) from (select PID from E:/txt/PRODUCT_SALE.txt group by PID)</pre>	/group data and then perform distinct and count

A1's result

_1
100000

✦ 14.11 Count distinct in each group after grouping



Use SQL to count distinct in each group after grouping.

Count the number of days with sales records for each product based on *PRODUCT_SALE.txt*.

Data in the file

ID	PID	DATE	QUANTITY	SID	
1211	10075052	2010-01-01	84	10225	
2474	10098045	2010-01-01	106	10591	
10576	10093980	2010-01-01	53	10720	
...					

✦ 14.11 Count distinct in each group after grouping



SPL script is as follows:

	A	B
1	<pre>\$select PID,count(distinct DATE) as days_with_sales from E:/txt/PRODUCT_SALE.txt group by PID</pre>	<pre>/count(distinct)+group</pre>
2	<pre>\$select PID,count(*) as days_with_sales from (select PID from E:/txt/PRODUCT_SALE.txt group by PID,DATE) group by PID</pre>	<pre>/group+group</pre>
3	<pre>\$select PID,count(*) as days_with_sales from (select distinct PID,DATE from E:/txt/PRODUCT_SALE.txt) group by PID</pre>	<pre>/distinct+group</pre>

A1's result

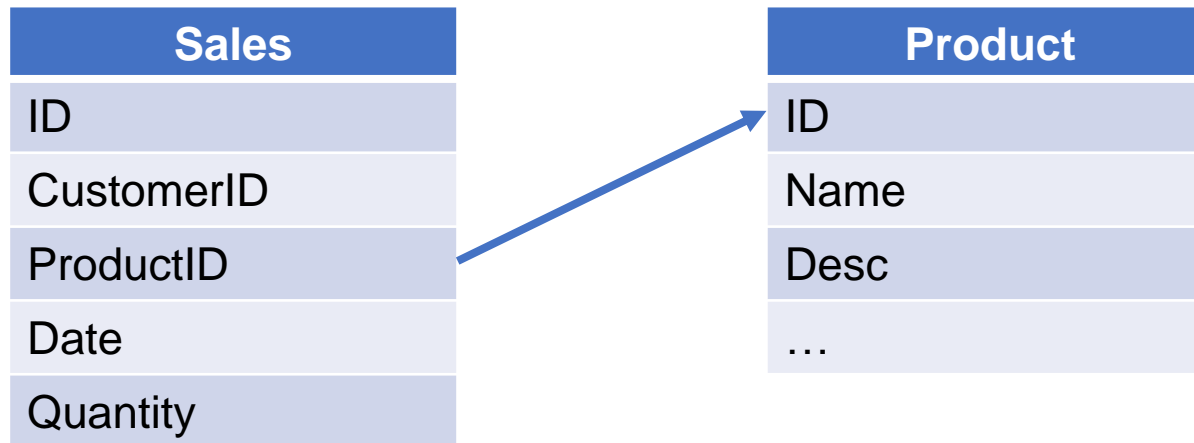
PID	days_with_sales
10000002	89
10000003	111
10000004	101
...	...

✦ 14.12 Join query over two text files



Use SQL to perform a join query over two text files.

Calculate the total sales amount of products with each sales quantity less than 10 based on *Sales.txt* and *Products.txt*.



✦ 14.12 Join query over two text files



SPL script is as follows:

	A	B
1	<pre>\$select sum(S.quantity*P.Price) as total from E:/txt/Sales.txt as S join E:/txt/Products.txt as P on S.productid=P.ID where S.quantity<=10</pre>	<pre>/join, filter, aggregate</pre>

A1's result

total
142740.180

✦ 14.13 Join query over multiple files



Use SQL to perform a join query over multiple files.

Find the employees of the HR Department in California based on *EMPLOYEE_J.txt*, *DEPARTMENT.txt* and *STATE.txt*.

✦ 14.13 Join query over multiple files



SPL script is as follows:

	A	B
1	<pre>\$select e.NAME as NAME from E:/txt/EMPLOYEE_J.txt as e join E:/txt/DEPARTMENT.txt as d on e.DEPTID=d.DEPTID join E:/txt/STATE.txt as s on e.STATEID=s.STATEID where d.NAME='HR' and s.NAME='California'</pre>	/Single level multi-foreign-key join

A1's result

NAME
Gabriel
Megan

✦ 14.14 Multi-level join query over multiple files



Use SQL to perform a multi-level join query over multiple text files.
Query employees in New York state whose manager is in California.

✦ 14.14 Multi-level join query over multiple files



SPL script is as follows:

	A	B
1	<pre>\$select e.NAME as ENAME from E:/txt/EMPLOYEE.txt as e join E:/txt/DEPARTMENT.txt as d on e.DEPT=d.NAME join E:/txt/EMPLOYEE.txt as emp on d.MANAGER=emp.EID where e.STATE='New York' and emp.STATE='California'</pre>	/Multi-level foreign key join

A1's result

NAME
Jessica
Alexis
Cameron
...

✦ 14.15 Using nested subquery



You can use nested SQL subqueries in SPL.

Find the department with the youngest manager based on *DEPARTMENT.txt* and *EMPLOYEE.txt*.

Data in the files:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
...

DEPTID	NAME	MANAGER
1	Administration	20
2	Finance	2
...

✦ 14.15 Using nested subquery



SPL script is as follows:

	A
1	<pre>\$select emp.DEPT as DEPT from E:/txt/DEPARTMENT.txt as dept join E:/txt/EMPLOYEE.txt emp on dept.MANAGER=emp.EID where emp.BIRTHDAY=(select max(BIRTHDAY) from (select emp1.BIRTHDAY as BIRTHDAY from E:/txt/DEPARTMENT.txt as dept1 join E:/txt/EMPLOYEE.txt as emp1 on dept1.MANAGER=emp1.EID))</pre>

A1's result

DEPT
Finance

✦ 14.16 Using common table expression (CTE)



Use SQL WITH clause to calculate data in a text file.

For example, find HR, R&D and sales department from *DEPARTMENT.txt*, and then calculate the number and average salary of female employees in these departments.

Text content

EMPLOYEE.txt

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
...

DEPARTMENT.txt

DEPTID	NAME	MANAGER
1	Administration	20
2	Finance	2
...

✦ 14.16 Using common table expression (CTE)



SPL script is as follows:

	A	B
1	<pre>\$with A as (select NAME as DEPT from E:/txt/DEPARTMENT.txt where NAME='HR' or NAME='R&D' or NAME='sales') select A.DEPT DEPT,count(*) NUM,avg(B.SALARY) AVG_SALARY from A left join E:/txt/EMPLOYEE.txt B on A.DEPT=B.DEPT where B.GENDER='F' group by A.DEPT</pre>	<pre>/with...as...</pre>

A1's result

DEPT	NUM	AVG_SALARY
HR	10	6300.0
R&D	11	8954.5
Sales	96	7250.0
...

✦ 14.17 Using command line to execute SQL



You can execute simple SQL in the command line to query a text file.

Calculate the average salary of each department based on the employee salary file.

Data in the file

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
...

✦ 14.17 Using command line to execute SQL



Use a cd command in command line to get into esProc/bin (where esprocx.exe is located) and execute the SQL statement using the following command: `.\esprocx -r "SQL statement"`

Command line content

```
.\esprocx -r "select DEPT,avg(SALARY) from E:/txt/EMPLOYEE.txt group by DEPT"
```

Execution result

```
E:\esproc\esProc\bin>.\esprocx -r "select DEPT,avg(SALARY) from E:/txt/EMPLOYEE.txt group by DEPT"
```

```
Administration 10000.0
Finance 7395.833333333333
HR 7263.1578947368425
Marketing 7409.090909090909
Production 7285.714285714285
R&D 8241.379310344828
Sales 7286.096256684492
Technology 7319.148936170212
```



Chapter 15

SPL
COOKBOOK

Using Excel data

✦ 15.1 Read xlsx data in simple format



Read data in an xlsx file. The first row in the file is the column title. Starting from the second row, each is a record.

	A	B	C	D	E	F	G
1	No	Name	Class	Sex	Chinese	Maths	English
2	110210	Lorry	5(1)	M	80	60	86
3	110211	Tom	5(1)	M	81	72	67
4	110212	Curry	5(1)	F	97	91	87
5	110213	Joan	5(1)	F	86	69	73

✦ 15.1 Read xlsx data in simple format



SPL script:

	A
1	<code>=file("scores.xlsx").xlsimport@t()</code>
2	<code>=file("scrores.txt").export@t(A1)</code>

A1 Open file "scores.xlsx" and import it as a table sequence; @t option indicates importing the first row as the title

A2 Write the table sequence in A1 into a text file; @t option means exporting the title first

The table sequence obtained in A1 is as follows:

Index	No	Name	Class	Sex	Chinese	Maths	English
1	110210	Lorry	5(1)	M	80	60	86
2	110211	Tom	5(1)	M	81	72	67
3	110212	Curry	5(1)	F	97	91	87
4	110213	Joan	5(1)	F	86	69	73

✦ 15.2 Read xlsx data with complex header



Read data in an xlsx file with complex header. The header of the file contains table name, item name and related information.

	A	B	C	D	E	F	G
1	Item Lists And Prices						
2	Project: Building			page 1/total 52			
3	No	Item Code	Item Name	Unit	Quantity	SumOfMoney(yuan)	
4						Price	Sum
5	1.1.2	NJSJ	Internal scaffolding	term			
6	1.1.2.1	11001004001	Internal scaffolding	term	1.00	1006577.54	1006577.54
7	1.1.2.1.1.1	A22-28	Steel pipe	100m ²	137.88	912.07	125756.21
8	1.1.2.1.1.2	A22-28	Base of internal scaffolding	100m ²	71.83	912.07	65513.99

✦ 15.2 Read xlsx data with complex header



SPL script:

	A
1	<code>=file("listsAndPrices.xlsx").xlsimport(;1,5)</code>
2	<code>=A1.rename(#1:No,#2:ItemCode,#3:ItemName,#4:Unit,#5:Quantity,#6:Price,#7:Sum)</code>

A1 Open the file and import data into a table sequence. The parameter "1,5" means reading data through to the end in the first sheet starting from line 5.

A2 Change the column names of table sequence in A1 to "No, ItemCode, ItemName, Unit, Quantity, Price, Sum".




The result table sequence is as follows:

Index	No	ItemCode	ItemName	Unit	Quantity	Price	Sum
1	1.1.2	NJSJ	Internal	term			
2	1.1.2.1	1.1001E+10	Internal	term	1	1006577.54	1006577.54
3	1.1.2.1.1.1	A22-28	Steel pipe	100m ²	137.88	912.07	125756.21
4	1.1.2.1.1.2	A22-28	Base of	100m ²	71.83	912.07	65513.99

✦ 15.3 Read free format xlsx data



Read data in a free format xlsx file. The first column of the file contains field names, which are followed by the free format data values on the right.

	A	B	C	D	E	F	G	
1	ID:	1						
2	Name:	Yin Zhang	Sex	F				
3	Position	Sales						
4	Birthday	1968-12-08						
5	Phone:	(010) 65559857						
7	Address:	No. 236 of Fuxingmen Beijing						
8	PostCode:	100098						
9								
10	ID:	2						
11	Name:	Wei Wang	Sex	M				
12	Position	Sales Manager						
13	Birthday	1962-02-19						
14	Phone:	(010) 65559482						
16	Address:	No. 890 of Luoma Garden Beijing						
17	PostCode:	109801						
18								
19	ID:	3						

✦ 15.3 Read free format xlsx data



Each employee's information occupies 9 rows. SPL reads such a file in the following way:

SPL script:

	A	B	C
1	=create(ID,Name,Sex,Position,Birthday,Phone,Address,PostCode)		
2	=file("Employee.xlsx").xlsopen()		
3	[C,C,F,C,C,D,C,C]	[1,2,2,3,4,5,7,8]	
4	for	=A3.(~/B3(#)).(eval(\$[A2.xlscell(]/~/")"))	
5		if len(B4(1))==0	break
6		>A1.record(B4)	
7		>B3.run(~+=9)	

✦ 15.3 Read free format xlsx data



A1 Create an empty table sequence with column names as "ID, Name, Sex, Position, Birthday, Phone, Address, PostCode"

A2 Open the Excel file

A3 Define the sequence of column numbers that command the cells containing employee information

B3 Define the sequence of row numbers that command the cells containing employee information

A4 Use for loop to read each employee's information

B4 A3.(~/B3(#)) first gets the sequence of the cell numbers for the current employee, and then read cell values as a sequence of employee information. The first cycle is [C1, C2, F2, C3, C4, D5, C7, C8], the second cycle is [C10, C11, F11, C12, C13, D14, C16, C17]... Add 9 to the row number each time. \$[A2.xlsxcell(), which is equivalent to "A2.xlsxcell(", represents a string.

B5 Judge whether the employee ID value is empty. If it is empty, exit the loop and end the operation.

B6 Append an employee's information at the end of table sequence in A1

B7 Add 9 to each member of the sequence of row numbers to read information of the next employee

The table sequence obtained in A1 is as follows:

Index	ID	Name	Sex	Position	Birthday	Phone	Address	PostCode
1	1	Yin Zhang	F	Sales	1968-12-8	(010)65559857	No. 236 of Fuxingmen Beijing	100098
2	2	Wei Wang	M	Sales Manager	1962-2-19	(010)65559482	No. 890 of Luoma Garden Beijing	109801
3	3	Fang Li	F	Market	1973-8-30	(010)65553412	No. 253 of Shaoyao Garden Beijing	198033
4	4	Jianjie He	M	Market Manager	1968-9-19	(010)65558122	No. 45 of Qianmen Street Beijing	198052

✦ 15.4 Read the crosstab in an xlsx file



Read the crosstab in an xlsx file.

	A	B	C	D	E	F	G	H								
1	Orders Statistics															
2	Type	Area	West	East	Center	North	South	Northwest	Southwest							
		Amount														
		Urgent express								20	70	1	97	23	2	35
		Unified parcel								25	89	1	148	39	3	27
		Federal cargo								15	79	52	108	29	2	23
		Air transport								5	1	12	1	1	9	6
		Cash on Delivery								8	2	4	1	6	7	9
		General express								32	41	36	48	26	22	18

✦ 15.4 Read the crosstab in an xlsx file



SPL script:

	A
1	<code>=file("cross.xlsx").xlsimport@t(;1,2)</code>
2	<code>=A1.rename(#1:Type)</code>
3	<code>=A2.pivot@r(Type;Area,Amount)</code>

The table sequence obtained in A3 is as follows:

Index	Type	Area	Amount
1	Urgent express	West	20
2	Urgent express	East	70
3	Urgent express	Center	1
4	Urgent express	North	97
5	Urgent express	South	23
6	Urgent express	Northwest	2
7	Urgent express	Southwest	35
8	Unified parcel	West	25
9	Unified parcel	East	89

A1 Open the file and import data as a table sequence.

A2 Since the first cell in the second row contains a picture, the read data is null and the first column doesn't have a title. Change the name of the first column to Type.

A3 Perform column to row transposition over the table sequence by grouping by type. @r options means to convert column data to row data. After conversion, the new column names are "Area" and "Amount".

✦ 15.5 Read the main & sub table in xlsx file



Read the main & sub table in an xlsx file. The xlsx file is composed of multiple worksheets. Each worksheet contains a main table and the sub table.

	A	B	C	D	E	F	G	H
1	Employee Information							
2	company	XX company				date:	1982-06-25	
3	Name	San Zhang	Sex	M	Birthday	1982-06-25	Nation	han
4	IDCard	510121198206253112			Phone	13612345678	Depart	Research
5	Home	No.25 Xisanqi			Marital	Married	Entry	2002-02
6	Family	Name	Relation	Workplace			Phone	
7		Zhou Zhang	farther	retired			15313231568	
8		Yin Hu	wife	XX company			13718826593	
9		Wuji Zhang	son	XX school				

✦ 15.5 Read the main & sub table in xlsx file



SPL script:

	A	B	C
1	=create(IDCard,Name,Sex,Birthday,Nation,Phone,Depart,Home,Marital,Entry)		
2	=create(IDCard,Name,Relation,Workplace,Phone)		
3	[B4,B3,D3,F3,H3,F4,H4,B5,F5,H5]		
4	=file("employee.xlsx").xlsopen()		
5	for A4	=A3.(eval(\$[A4.xlscell(]/~/",\""/A5.stname/\"/\"))	>A1.record(B5)
6		=A4.xlsimport@t(Family,Name,Relation,Workplace,Phone;A5.stname,6)	
7		=B6.rename(Family:IDCard)	>B7.run(IDCard=B5(1))
8		>A2.insert@r(0:B7)	

A1 Create an empty table sequence with column names "IDCard, Name, Sex, Birthday, Nation, Phone, Depart, Home, Marital, Entry" to store the employee information in the main table

A2 Create an empty table sequence with column names "IDCard, Name, Relation, Workplace, Phone" to store the information of employee family members in the sub table

A3 Define the sequence of cells holding employee information in the main table

A4 Open the Excel file

A5 Read each worksheet of the Excel file in loop

✦ 15.5 Read the main & sub table in xlsx file



- B5** Read employee information as a sequence
- C5** Save the employee information read by B5 in A1's table sequence
- B6** Read the employee family member information from row 6 onward; read only the specified "Family, Name, Relation, Workplace, Phone" columns.
- B7** Change the name of family column of B6's table sequence to IDCard
- C7** Assign values of IDCard column in employee information to IDCard column in B7's table
- B8** Append employee family member information in B7 to A2's table sequence

The table sequence obtained in A1 is as follows:

Index	IDCard	Name	Sex	Birthday	Nation	Phone	Depart	Home	Marital	Entry
1	510121198206253000	San Zhang	M	1982-6-25	han	13612345678	Research	No.25 Xisanqi	married	2002-2-1
2	110114198907286000	Si Li	M	1989-7-28	han	13818022624	Sales	No.302 Lifeng Garden	free	2008-6-1
3	310503198803243000	Xiaoyu Zhao	F	1988-3-24	miao	13852416325	HR	No.501 Yuquyuan	married	2005-1-1

The table sequence obtained in A2 is as follows:

Index	IDCard	Name	Relation	Workplace	Phone
1	510121198206253000	Zhou Zhang	farther	retired	15313231568
2	510121198206253000	Yin Hu	wife	XX company	13718826593
3	510121198206253000	Wuji Zhang	son	XX school	
4	110114198907286000	Dasuan Li	farther	XX trade company	13625689532
5	110114198907286000	Haixia Liu	mother	XX supermarket	13924689512
6	310503198803243000	Darong Luo	husband	XX software company	13598325647
7	310503198803243000	Xiaolu Luo	daughter	XX primary school	

✦ 15.6 Read big xlsx file



Read data in a big xlsx file and output it. The amount of data in the file is too large to be read into the memory at one time.

	A	B	C	D	E
1	ID	Company	Area	OrderDate	Amount
123661	10251	Qiangu	East	2012-07-08	624.9
123662	10252	Fuxing	West	2012-07-09	3059.49
123663	10253	Shiyi	North	2012-07-10	1428
123664	10254	Haotian	Center	2012-07-11	545.3

✦ 15.6 Read big xlsx file



SPL script:

	A
1	<code>=file("orders.xlsx").xlsimport@tc()</code>
2	<code>=file("orders.txt").export@t(A1)</code>

This example is very similar to 15.1, except that the @c option is used in A1 to read data with a cursor. Only Excel files in xlsx format can be read using cursor.

A1 Open *orders.xlsx* file and import it as a cursor. @t option indicates that the first row of the file is the column title, and @c option indicates returning a cursor.

A2 Export the cursor data in A1 to the text file *orders.txt*. @t option means to export the title first.

✦ 15.7 Write a data table to xlsx file



Read a data table in a text file and export it to an xlsx file.
Write the data table and its column headers stored in text file *orders.txt* to Excel file *orders.xlsx*.

✦ 15.7 Write a data table to xlsx file



SPL script:

	A
1	<code>=file("orders.txt":"UTF-8").import@t()</code>
2	<code>=file("orders.xlsx").xlsexport@t(A1)</code>

A1 Import the orders table in text format.

A2 Export A1's data to the *orders.xlsx* file (which will be automatically created if the file does not exist). Since no field is specified through parameter in `xlsexport` function, all fields of A1 will be exported. As no target worksheet is specified, data is exported to Sheet1. The function uses `@t` option to export the field names to the first row.

The exported Excel file:

	A	B	C	D	E	F
1	ID	Company	Area	OrderDate	Amount	Phone
2	10248	Shantai	North	2012-07-04	428	(030) 26471510
3	10249	Dongdiwar	East	2012-07-05	1842	(0251) 1031259
4	10250	Shiyi	North	2012-07-08	1523.5	(0211) 5550091
5	10251	Qiangu	East	2012-07-08	624.95	(071) 8325486
6	10252	Fuxing	West	2012-07-09	3559.5	(030) 23672220
7	10253	Shiyi	North	2012-07-10	1428	(0211) 5550091
8	10254	Haotian	Center	2012-07-11	545.4	(030) 30076545
9	10255	Yongda	North	2012-07-12	2450	(089) 7034214

✦ 15.8 Append data table to xlsx file



Append a data table to an xlsx file. The new data is added after the existing data in the file. Append the data table stored in text file *aday.txt* to Excel file *orders.xlsx*.

✦ 15.8 Append data table to xlsx file



SPL script:

	A
1	<code>=file("aday.txt":"UTF-8").import@t()</code>
2	<code>=file("orders.xlsx").xlsexport@a(A1)</code>

If the Excel file already exists, use @a option to append data to the original file. In this case, @t option is not support.

✦ 15.9 Write data table to different worksheets of an xlsx file



Export a data table to different sheets of an xlsx file.

Export data of Shantai company in text file *orders.txt* to worksheet Shantai in Excel file *orders.xlsx*.

✦ 15.9 Write data table to different sheets of an xlsx file



SPL script:

	A
1	<code>=file("orders.txt":"UTF-8").import@t()</code>
2	<code>=A1.select(Company=="Shantai")</code>
3	<code>=file("orders.xlsx").xlsexport@t(A2,ID,Company,OrderDate:Date,Amount:Money;"Shantai")</code>

A2: Filter A1's table sequence

A3: Export A2 to *orders.xlsx*. Only four fields, ID, Company, OrderDate and Amount, are exported; and rename OrderDate as Date, rename Amount as Money. Data is exported to a **sheet named Shantai**.

The exported Excel file:

	A	B	C	D
1	ID	Company	Date	Money
2	10248	Shantai	2012-07-04	428
3	10274	Shantai	2012-08-06	529
4	10295	Shantai	2012-09-02	120
5				

Sheet1 Shantai

✦ 15.10 Export a large amount of data to xlsx file



When exporting a large amount of data to an xlsx file, the data can not be entirely read into memory, so we use cursor to read data row by row

Write data of the big text file *big.txt* to Excel file *big.xlsx*.

When exporting data with a cursor, you need to use @s option to do a stream-style export to make sure that the memory receives only a small amount of data.

✦ 15.10 Export a large amount of data to xlsx file



SPL script:

	A
1	=file("big.txt":"UTF-8").cursor@t()
2	=file("big.xlsx").xlsexport@st(A1)

The exported Excel file:

	A	B	C	D	E
1	ID	Company	Area	OrderDate	Amount
123658	10248	Shantai	North	2012-07-04	428
123659	10249	Dongdiwang	East	2012-07-05	1842
123660	10250	Shiyi	North	2012-07-08	123.49
123661	10251	Qiangui	East	2012-07-08	624.9
123662	10252	Fuxing	West	2012-07-09	3059.49
123663	10253	Shiyi	North	2012-07-10	1428
123664	10254	Haotian	Center	2012-07-11	545.3
123665					

In this example, 123663 records are exported. This method can export hundreds of millions of records. However, an Excel worksheet can only hold 1048576 rows of data at most, so when the exported data reaches a million rows, a new worksheet is needed to store data.

✦ 15.11 Sort after join



Sort two joined tables by specified expression.

Based on *Indicators.xlsx*, get the indicators of U.S. and China that have a large gap between the two countries.

	A	B	C	D	E	F	G	H
1	Indicator	Last	Previous	Min	Max	Unit	Frequency	Range
2	Govern--ment Debt: % of GDP	105.7	107.3	31.8	107.3	%	Yearly	1969 - 2017
3		2017	2016	1974	2016			Updated on 2018---03-28
4	Business Confi--dence: Net Balance	18.6	21.6	-41.2	55	% Point	Monthly	Jan 1948 - Mar 2018
5		Mar-18	Feb-18	May-80	Jul-50			Updated on 2018---04-02
6	Foreign Direct Inves--tment	49,814.00	105,653.00	-75,269.00	246,224.00	USD mn	Quarterly	Mar 1960 - Dec 2017
7		Dec-17	Sep-17	Mar-14	Mar-15			Updated on 2018---03-21
8	Total Imports Growth	10.8	7.3	-35.4	33.2	%	Monthly	Jan 1990 - Feb 2018
9		Feb-18	Jan-18	May-09	May-10			Updated on Mar 2018
10	Money Supply M2	13,858,400.00	13837800.00	286,600.00	13,858,400.00	USD mn	Monthly	Jan 1959 - Feb 2018
11		Feb-18	Jan-18	Jan-59	Feb-18			Updated on 2018---03-29
12	Forecast: Gover--nment Revenue	7,612.00	7,322.72	3,270.25	7,612.00	USD bn	Yearly	2001 - 2022
13		2022	2021	2002	2022			Updated on 2017---10-10
14	Motor Vehicle Produ--ction	11,189,985.00	12,180,301.00	5,377,687.00	12,279,582.00	Unit	Semia--nnual	Dec 2001 - Dec 2017
15		Dec-17	Dec-16	Jun-12	Dec-02			Updated on 2018---03-08
16	Tele--ensity: Fixed Line	37.09	38.4	26.44	67.64	Number	Yearly	1960 - 2016
17		2016	2015	1960	2000			Updated on 2017
18	Natural Gas: Exports	65,542.00	50,502.00	4,000.00	65,542.00	Cub m mn	Yearly	1995 - 2016
19		2016	2015	1996	2016			Updated on 2017---06-13
20	Real GDP Growth	2.6	2.3	-4.1	13.4	%	Quarterly	Mar 1948 - Dec 2017
21		Dec-17	Sep-17	Jun-09	Dec-50			Updated on Mar 2018
22	Labour Produ--ctivity Growth	1.36	0.9	-2.65	9.72	%	Quarterly	Mar 1949 - Dec 2017
23		Dec-17	Sep-17	Mar-74	Dec-50			Updated on 2018---03-28
24	Domestic Credit Growth	5.1	4.9	-2.4	10	%	Quarterly	Dec 2002 - Dec 2016
25		Dec-16	Sep-16	Jun-10	Mar-08			Updated on 2018---03-19
26		146.0	138.2	26.6	152			1975 - 2016

✦ 15.11 Sort after join



SPL script is as follows, where A.sort() function is used to sort the data:

	A	B
1	=file("Indicators.xlsx").xlsopen()	/Open the Excel file
2	=A1.xlsimport@t(Indicator,Last).select(Indicator!=null)	/Import the first sheet (USA) and select records whose <i>Indicator</i> value is not null
3	=A1.xlsimport@t(Indicator,Last;"China").select(Indicator!=null)	/Import the sheet containing indicators of China and select records whose <i>Indicator</i> value is not null
4	=A2.join(Indicator,A3:Indicator,Last:'China').rename(Last:'United States')	/Join tables in A2 and A3 by <i>Indicator</i> field
5	=A4.sort@z(max('United States','China') / min('United States','China'))	/Sort by the ratio of indicators (the larger one to the smaller one) of the US and China in descending order, where sort function is used to sort the data and @z option means descending order

A5	Indicator	United States	China
	Foreign Exchange Reserves: % of GDP	0.22	25.6
	Foreign Exchange Reserves: Months of Import	0.2	22.7
	Imports: Television	2.124857134E7	198972.78
	Foreign Exchange Reserves	44123	3134482

✦ 15.12 Specify display attributes



Sometimes we want to be able to specify the format for the generated excel file (such as font, color, background color, alignment, etc.). To do this, just build the Excel file template in advance, define the format, and then export data to the file.

As shown below, write the table name in the first row of sheet1 in the *orders.xlsx* file, the column names in the second row, and define some style attributes for the table name and each column. The first, third and fourth columns is centrally aligned, the second column is left aligned, and the fifth column the right aligned. The display format of the fourth column is "yyyy-mm-dd", and that of the fifth column is "#,###.00".

	A	B	C	D	E
1	Product Orders				
2	col1	col2	col3	col4	col5
3					

✦ 15.12 Specify display attributes



SPL script:

	A
1	=file("orders.txt":"UTF-8").import@t()
2	=file("orders.xlsx").xlsexport@t(A1)

The SPL script is the same as that for the previous example in 15.7. The exported result is shown below. When exporting to an existing file, the last non empty line of the file will be overwritten to be used as the header. The style attributes defined in the original file will be retained (not supported in stream-style export of a large amount of data).

The exported Excel file:

	A	B	C	D	E
1	Product Orders				
2	ID	Company	Area	OrderDate	Amount
3	10257	Yuandong	East	2012-07-16	1,109.00
4	10258	Zhengren	East	2012-07-17	1,604.00
5	10259	Sanjie	West	2012-07-18	100.00
6	10260	Jingmi	North	2012-07-19	1,461.75
7	10261	Lange	South	2012-07-19	440.00
8	10262	Xueren	Center	2012-07-22	583.20

✦ 15.13 Fill in the specified cell or area of an xlsx file



SPL also provides a way to read or write a specified cell or a block of cells in an Excel file. You can fill in data to an Excel of a fixed format, such as the following Excel file:

The exported Excel file:

	A	B	C	D	E	F	G	H	I	J	K	L
1	General Information of Fund Company											
2	Company								Year		Season	
3	Net assets				Total assets				Total assets(share)			
4	Inherent capital investment	Deposit	National debt	Closed-end funds		Open-ended Funds		Buy-back securities	Negotiable deposit	Policy financial bonds	Central Bank Bills	Other
5				Total share	Our share	Total share	Our share					
6												
7	Employee Holding Fund	Investors	Total share	Not our company manages funds		Our company manages funds						
8				Investors	Total share	Investors	Total share					
9												
10	Employee turnover	Total employee		New employee of this season				Leaving employee of this season				
11												

	A	B	C	D	E	F	G	H	I	J	K	L
1	General Information of Fund Company											
2	Company	Mengniu funds company							Year	2012	Season	3
3	Net assets	58.2			Total assets		364		Total assets(share)		300	
4	Inherent capital investment	Deposit	National debt	Closed-end funds		Open-ended Funds		Buy-back securities	Negotiable deposit	Policy financial bonds	Central Bank Bills	Other
5				Total share	Our share	Total share	Our share					
6			8.5	50	200	100	400	200	182.6	76.3	43.7	28.5
7	Employee Holding Fund	Investors	Total share	Not our company manages funds		Our company manages funds						
8				Investors	Total share	Investors	Total share					
9					120	1.07	30	0.27	90	0.8		
10	Employee turnover	Total employee		New employee of this season				Leaving employee of this season				
11		154		6				4				

✦ 15.13 Fill in the specified cell or area of an xlsx file



SPL script:

	A	B	C	D	E	F
1	Mengniu funds company	2012	3	58.2	364	300
2	8.5	50	200	100	400	200
3	182.6	76.3	43.7	28.5	16.4	
4	120	1.07	30	0.27	90	0.8
5	154	6	4			
6	=file("result.xlsx")		=A6.xlsopen()			
7	=C6.xlscell("B2",1;A1)		=C6.xlscell("J2",1;B1)		=C6.xlscell("L2",1;C1)	
8	=C6.xlscell("B3",1;D1)		=C6.xlscell("G3",1;E1)		=C6.xlscell("K3",1;F1)	
9	=C6.xlscell("B6",1;[A2:F2].concat("\t"))			=C6.xlscell("H6",1;[A3:E3].concat("\t"))		
10	=C6.xlscell("B9",1;[A4:F4].concat("\t"))			=C6.xlscell("B11",1;[A5:C5].concat("\t"))		
11	=A6.xlswrite(C6)					

It is assumed that the data to be filled in has been calculated (in the first 5 rows). The first six cells to be filled in the sample table are all independent, so only one cell can be filled in at a time. The sixth row has cells that can be filled in consecutively. We can concatenate the data to be filled in into a string separated by \t, which can be filled in starting from the specified cell in order. After all data is filled in, write the Excel object in C6 back to the *result.xlsx* file.

✦ 15.14 Export row-style report to xlsx file



Export a row-style report of complex format to an xlsx file.

Export *orders list* to xlsx file with the following requirements: show the background color of the data rows alternately in two colors; display cells with the order amount greater than 2000 in red, and those with the order amount less than 500 in green.

	A	B	C	D	E
1	Orders List				
2	ID	Company	Area	Orderdate	Amount
3	10248	Shantai	North	2012-07-04	428.00
4	10249	Dongdiwang	East	2012-07-05	1842.00
5	10250	Shiyi	North	2012-07-08	1523.50
6	10251	Qiangu	East	2012-07-08	624.95
7	10252	Fuxing	West	2012-07-09	3559.50
8	10253	Shiyi	North	2012-07-10	1428.00
9	10254	Haotian	Center	2012-07-11	545.40
10	10255	Yongda	North	2012-07-12	2450.00

✦ 15.14 Export row-style report to xlsx file



Implementation steps:

Open RaqReport designer and create a new report template "orders.rpx", the screenshot is as follows:

	A	B	C	D	E
1	Orders List				
2	ID	Company	Area	Orderdate	Amount
3	=ds1.select()	=ds1.Company	=ds1.Area	=ds1.Orderdate	=ds1.Amount

Suppose the name of the dataset passed from esProc is ds1, and the data rows start from the third row of the report. For the usage of the RaqReport, please refer to the RaqReport tutorial.

Select all the cells in the third row and write the background color expression as `if(row()%2==0,-853778,-1)` to specify the two background colors to be displayed alternately.

Select the last cell in the third row, specify the display format as `#.00`, write the foreground color expression as `if (value()>2000, -65536, if (value()<500, -16711936, -16777216))` to display different font colors according to different amounts.

✦ 15.14 Export row-style report to xlsx file



SPL script:

	A
1	=file("orders.txt":"UTF-8").import@t()
2	>report_config("E:\\report\\raqsoftConfig.xml")
3	=report_open("orders.rpx")
4	=report_run(A3;A1:"ds1")
5	=report_exportXls@x(A3,"rpt.xlsx")

A1 Read the table sequence to be exported;

A2 Configure the report environment, including mainly the report main directory and license file. You can copy the licensing XML file from the report installation directory;

A3 Open the report template;

A4 Pass the table sequence in A1 as dataset ds1 to report object A3 for calculation;

A5 Export the calculated report object A3 into an Excel file.

The exported Excel file:

	A	B	C	D	E
1	Orders List				
2	ID	Company	Area	Orderdate	Amount
3	10248	Shantai	North	2012-07-04	428.00
4	10249	Dongdiwang	East	2012-07-05	1842.00
5	10250	Shiyi	North	2012-07-08	1523.50
6	10251	Qiangui	East	2012-07-08	624.95
7	10252	Fuxing	West	2012-07-09	3559.50
8	10253	Shiyi	North	2012-07-10	1428.00
9	10254	Haotian	Center	2012-07-11	545.40
10	10255	Yongda	North	2012-07-12	2450.00

✦ 15.15 Export multi-level grouped report to xlsx file



Create a multi-level grouped report using RaqReport and export it to an xlsx file.

Export a multi-level grouped *Orders Statistical Table* to an xlsx file according to the specified format, such as follows:

ID	Company	Area	OrderDate	Amount
10248	Shantai	North	2012-07-04	428.0
10249	Dongdiwang	East	2012-07-05	1842.0
10250	Shiyi	North	2012-07-08	1523.50
10251	Qiangu	East	2012-07-08	624.95
10252	Fuxing	West	2012-07-09	3559.50
10253	Shiyi	North	2012-07-10	1428.0
10254	Haotian	Center	2012-07-11	545.40
10255	Yongda	North	2012-07-12	2450.0



	A	B	C	D	E
1	Orders Statistical Table				
2	Area	Company	ID	Orderdate	Amount
3	Center	Haotian	10254	2012-07-11	545.40
4			Total Amount		545.40
5		Total Amount			
6	East	Dongdiwang	10249	2012-07-05	1842.00
7			Total Amount		1842.00
8		Qiangu	10251	2012-07-08	624.95
9	Total Amount		624.95		
10	Total Amount				2466.95
11	North	Shantai	10248	2012-07-04	428.00
12			Total Amount		428.00
13		Shiyi	10250	2012-07-08	1523.50
14			10253	2012-07-10	1428.00
15			Total Amount		2951.50
16		Yongda	10255	2012-07-12	2450.00
17	Total Amount		2450.00		
18	Total Amount				5829.50

✦ 15.15 Export multi-level grouped report to xlsx file



Implementation steps:

Open RaqReport designer and create a new report template "orders_group.rpx", the screenshot is as follows:

	A	B	C	D	E
1	Orders Statistical Table				
2	Area	Company	ID	Orderdate	Amount
3	=ds1.group(↓	=ds1.group(Company: ↓	=ds1.select(↓	=ds1.Orderdate	=ds1.Amount
4				Total Amount	=sum(E3{ })
5				Total Amount	=sum(E3{ })

✦ 15.15 Export multi-level grouped report to xlsx file



SPL script:

	A
1	=file("orders.txt":"UTF-8").import@t()
2	>report_config("E:\\report\\raqsoftConfig.xml")
3	=report_open("orders_group.rpx")
4	=report_run(A3;A1:"ds1")
5	=report_exportXls@x(A3,"rpt.xlsx")

Similar to the previous example, the dataset passed to the report is named ds1, which is grouped by area in A3 and by company name in B3, and where the order details are displayed in C3, D3 and E3. The total order amount of each company is calculated in E4, and the total order amount of each area is calculated in E5.

The exported Excel file:

	A	B	C	D	E
1	Orders Statistical Table				
2	Area	Company	ID	Orderdate	Amount
3	Center	Haotian	10254	2012-07-11	545.40
4			Total Amount		545.40
5		Total Amount			
6	East	Dongdiwang	10249	2012-07-05	1842.00
7			Total Amount		1842.00
8		Qianggu	10251	2012-07-08	624.95
9			Total Amount		624.95
10	Total Amount				2466.95
11	North	Shantai	10248	2012-07-04	428.00
12			Total Amount		428.00
13		Shiyi	10250	2012-07-08	1523.50
14			10253	2012-07-10	1428.00
15			Total Amount		2951.50
16		Yongda	10255	2012-07-12	2450.00
17			Total Amount		2450.00
18	Total Amount				5829.50

✦ 15.16 Export crosstab report to xlsx file



Create a crosstab report using RaqReport and export it to an xlsx file.

Export a *Orders Cross Statistical Table* to an xlsx file according to the specified format, such as follows:

ID	Company	Area	OrderDate	Amount
10248	Shantai	North	2012-07-04	428.0
10249	Dongdiwang	East	2012-07-05	1842.0
10250	Shiyi	North	2012-07-08	1523.50
10251	Qiangu	East	2012-07-08	624.95
10252	Fuxing	West	2012-07-09	3559.50
10253	Shiyi	North	2012-07-10	1428.0
10254	Haotian	Center	2012-07-11	545.40
10255	Yongda	North	2012-07-12	2450.0



	A	B	C	D	E	
1	Orders Cross Statistical Table					
2	<div>Area Year Amount</div>	2012	2013	2014	Total	
3		Center	1715.40	.00	.00	1715.40
4		East	54280.68	24591.52	13671.02	92543.22
5		North	88467.49	58859.06	34609.72	181936.27
6		South	13088.15	793.43	6475.80	20357.38
7		West	15932.44	22784.80	12995.14	51712.38
8		Total	173484.16	107028.81	67751.68	348264.66

◆ 15.16 Export crosstab report to xlsx file



Implementation steps:

Use RaqReport designer to edit report template "orders_cross.rpx", as shown on the right:

SPL script:

	A
1	=file("orders.txt":"UTF-8").import@t()
2	>report_config("E:\\report\\raqsoftConfig.xml")
3	=report_open("orders_cross.rpx")
4	=report_run(A3;A1:"ds1")
5	=report_exportXls@x(A3,"rpt.xlsx")

Similar to the previous example, the dataset passed to the report is named ds1, which is grouped by year of orders in B2 and by area in A3. B3 counts the total order amount of each subgroup.

	A	B	C
1	Orders Cross Statistical Table		
2	Area \ Year Amount	=ds1.group(year(Ord	Total
3	=ds1.group(Area	=ds1.sum(Amount)	=sum(B3{})
4	Total	=sum(B3{})	=sum(B3{})

The exported Excel file:

	A	B	C	D	E
1	Orders Cross Statistical Table				
2	Area \ Year Amount	2012	2013	2014	Total
3	Center	1715.40	.00	.00	1715.40
4	East	54280.68	24591.52	13671.02	92543.22
5	North	88467.49	58859.06	34609.72	181936.27
6	South	13088.15	793.43	6475.80	20357.38
7	West	15932.44	22784.80	12995.14	51712.38
8	Total	173484.16	107028.81	67751.68	348264.66

✦ 15.17 Combine multiple xlsx files of same structure



Read multiple xlsx files of same structure, merge them and export the data to one xlsx file.
Combine data in all xlsx files under E:/work/excel/ directory and export the result to *total.xlsx*.

✦ 15.17 Combine multiple xlsx files of same structure



SPL script:

	A	B	C
1	>dir="E:/work/excel/"	=file(dir+"total.xlsx")	
2	=directory(dir+"*.xlsx")		
3	for A2	=file(dir+A3).xlsimport@t()	
4		If #A3==1	=B1.xlsxexport@t(B3)
5		else	=B1.xlsxexport@a(B3)

A1 Define the directory where the Excel files to be processed are located;

B1 Open the target file to which data is exported after combination;

A2 List the names of all xlsx files in the directory;

A3 Loop through the listed files;

B3 Read the first Excel file into a table sequence;

B4-C5 Export the table sequence in B3, with column header exported for the first time.

Note: Here we do the illustration using files of simple format. For files of complicated formats, please refer to the parsing methods explained in previous chapters.

✦ 15.18 Split an xlsx file and export it to different xlsx files



Group data in an xlsx file and export it to different xlsx files.
Export data of different kinds of parts stored in *orders.xlsx* to a set of corresponding xlsx files.

✦ 15.18 Split an xlsx file and export it to different xlsx files



SPL script:

	A	B
1	>dir="E:/work/excel/"	
2	=file("orders.xlsx").xlsimport@t()	=A2.group(partName)
3	for B2	=file(dir+A3(1).partName+".xlsx").xlsexport@t(A3)

A2 Read all data;

B2 Group by partName;

A3 Export the order information of each part in loop;

B3 Export the information of each part to a file named after the part.

Note: Here we do the illustration using files of simple format. For files of complicated formats, please refer to the export methods explained in previous sections.



Chapter 16

SPL
COOKBOOK

Using JSON and XML data

✦ 16.1 Import single-layer JSON file



Import data from a single-layer JSON file.

The following is product information of JSON format:

```
[{"PRODUCT_ID":1,"PRODUCT_NAME":"Apple Juice",  
"SUPPLIER_ID":2,"CATEGORY_ID":1, ...},  
{"PRODUCT_ID":2,"PRODUCT_NAME":"Milk",  
"SUPPLIER_ID":1,"CATEGORY_ID":1, ...},  
{"PRODUCT_ID":3,"PRODUCT_NAME":"Tomato sauce",  
"SUPPLIER_ID":1,"CATEGORY_ID":2, ...},  
{"PRODUCT_ID":4,"PRODUCT_NAME":"Salt",  
"SUPPLIER_ID":2,"CATEGORY_ID":2, ...},  
...]
```

✦ 16.1 Import single-layer JSON file



SPL only needs a simple one-line script to import a JSON file:

```
=json(file("product.json").read())
```

Execution result is as follows:

PRODUCT_ID	PRODUCT_NAME	SUPPLIER_ID	CATEGORY_ID	...
1	Apple Juice	2	1	...
2	milk	1	1	...
3	Tomato sauce	1	2	...
4	salt	2	3	...
...

✦ 16.2 Import multi-layer JSON file with same-structure detailed data



Import data from a two-layer JSON file where the detailed data has a consistent structure.

The following is the order information of JSON format. It has two layers: the first layer is the country and area, and the second is the detailed data.

Import orders from north and south China in 2013.

```
[{"COUNTRY":"China","AREA":"Northeast China","ORDERS":[
{"ORDER_ID":10252,"CUSTOMER_ID":"SUPRD", "ORDER_DATE":2012-07-11, ...},
{"ORDER_ID":10318,"CUSTOMER_ID":"ISLAT", "ORDER_DATE":2012-07-25, ...},
...]},
{"COUNTRY":"China","AREA":"East China","ORDERS":[
{"ORDER_ID":10249,"CUSTOMER_ID":"TOMSP", "ORDER_DATE":2012-07-05, ...},
{"ORDER_ID":10251,"CUSTOMER_ID":"VICTE", "ORDER_DATE":2012-07-08, ...},
...]},
...]
```

✦ 16.2 Import multi-layer JSON file with same-structure detailed data



Define parameters Country, Area and Year. You just need to modify the corresponding parameter value instead of the SPL script when importing a different country, area or year later. Note that the Area value is a sequence, so that the data of multiple areas can be read at the same time. As shown below:

Name	Value
Country	China
Area	[North China, South China]
Year	2013

✦ 16.2 Import multi-layer JSON file with same-structure detailed data



The SPL script is as follows:

	A	B
1	<code>=json(file("orders.json").read())</code>	<code>=A1.select(COUNTRY==Country && Area.contain(AREA))</code>
2	<code>=B1.news(ORDERS;COUNTRY, AREA,{B1.ORDERS.fname().concat@c()})</code>	<code>=A2.select(year(ORDER_DATE)==Year)</code>

✦ 16.2 Import multi-layer JSON file with same-structure detailed data



```
=json(file("orders.json").read())
```

First, import the multi-layer JSON file:

COUNTRY	AREA	ORDERS
China	Northeast China	[[10252,SUPRD,4,...],[10315,ISLAT,4,...],...]
China	East China	[[10249,TOMSP,6,...],[10251,VICTE,3,...],...]
China	Central China	[[10254,CHOPS,5,...],[10265,BLONP,2,...],...]
China	North China	[[10248,VINET,5,...],[10250,HANAR,4,...],...]
China	South China	[[10287,RICAR,8,...],[10296,LILAS,6,...],...]



ORDER_ID	CUSTOMER_ID	EMPLOYEE_ID	...
10287	RICAR	8	...
10296	LILAS	6	...
...

✦ 16.2 Import multi-layer JSON file with same-structure detailed data



The Year and Area fields are in the first layer, so we can directly select the data of North China and South China.

```
=A1.select(COUNTRY==Country && Area.contain(AREA))
```

The result is as follows:

COUNTRY	AREA	ORDERS
China	North China	[[10248,VINET,5,...],[10250,HANAR,4,...],...]
China	South China	[[10287,RICAR,8,...],[10296,LILAS,6,...],...]

✦ 16.2 Import multi-layer JSON file with same-structure detailed data



```
=B1.news(ORDERS;COUNTRY, AREA,{B1.ORDERS.fname().concat@c()})
```

Generate a table sequence with the filtering result, which contains fields including country, area, order_id, etc. The result is as follows:

COUNTRY	AREA	ORDER_ID	CUSTOMER_ID	EMPLOYEE_ID	ORDER_DATE
China	North China	10248	VINET	5	2012-07-04
China	North China	10250	HANAR	4	2012-07-08
China	North China	10253	HANAR	3	2012-07-10
China	North China	10255	RICSU	9	2012-07-12

Here news() function uses macro in one of its parameters. A macro uses \${} to enclose the expression. SPL will calculate the macro expression first, and then replace \${} with the calculated result as a string value. Actually A2 is equivalent to =B1.news(ORDERS;COUNTRY, AREA, ORDER_ID, CUSTOMER_ID, EMPLOYEE_ID, ORDER_DATE, ...)

✦ 16.2 Import multi-layer JSON file with same-structure detailed data



Finally, select records where the order year is 2013 from the table sequence.

```
=A2.select(year(ORDER_DATE)==Year)
```

The final result is as follows:

COUNTRY	AREA	ORDER_ID	CUSTOMER_ID	EMPLOYEE_ID	ORDER_DATE
China	North China	10402	ERNSH	8	2013-01-02
China	North China	10403	ERNSH	4	2013-01-03
China	North China	10404	MAGAA	2	2013-01-03
China	North China	10407	OTTIK	2	2013-01-07

✦ 16.3 Import multi-layer JSON file with different-structure detailed data



Read data from a multi-layer JSON file where the detailed data has an inconsistent structure.

The detailed data of a JSON file may be of different structures because of the complexity of data sources. In the following sales data: the first layer takes year and month as the dimension, the second layer takes country as the dimension, and the third layer is detailed data. But in the detailed data, due to different sales channels, the data structure is not completely consistent.

Import sales data for 2017 and 2018 in the US and Canada.

```
[{"YEAR":2016,"MONTH":1,"SALES":  
  [{"COUNTRY":"Germany","SALES":  
    [{"ORDERNUMBER":10101,"QUANTITYORDERED":25,"PRICEEACH":100,"ORDE  
RLINENUMBER":4,"SALES":3782,"ORDERDATE":"1/9/2016 0:00", ...}, ...], ...},  
 {"YEAR":2016,"MONTH":2,"SALES":  
  [{"COUNTRY":"Denmark","SALES":  
    [{"ORDERNUMBER":10105,"QUANTITYORDERED":50,"PRICEEACH":100,"ORDE  
RLINENUMBER":2,"SALES":7208,"ORDERDATE":"2/11/2016 0:00", ...}, ...], ...},  
 ...]
```

✦ 16.3 Import multi-layer JSON file with different-structure detailed data



First we need to determine the structure of the detailed data. In this example, we want to list all the fields. If the detailed data does not contain a certain field, it is set to null. For example, the addressline2 field is missing from the following data:

YEAR	COUNTRY	ORDERNUMBER	ADDRESSLINE1	ADDRESSLINE2
2017	USA	10353	2440 Pompton St.	
2017	USA	10352	16780 Pompton St.	

We'll define two parameters - Year and Country, for the convenience of calculation.

Name	Value
Year	[2017, 2018]
Country	[USA, Canada]

✦ 16.3 Import multi-layer JSON file with different-structure detailed data



The SPL script is as follows:

	A	B
1	<code>=json(file("sales.json").read())</code>	<code>=A1.select(Year.contain(YEAR))</code>
2	<code>=B1.news(SALES;YEAR,MONTH,COUNTRY,SALES)</code>	<code>=A2.select(Country.contain(COUNTRY))</code>
3	<code>for B2</code>	<code>=A3.SALES.fname()&B3</code>
4	<code>=B2.news(SALES; YEAR, COUNTRY,{B3.concat@c()})</code>	

✦ 16.3 Import multi-layer JSON file with different-structure detailed data



```
=json(file("sales.json").read())
```

```
=A1.select(Year.contain(YEAR))
```

First, import the multilayer JSON file. Since the Year field is in the first layer, we can directly select the data of 2017 and 2018:

YEAR	MONTH	SALES
2017	1	[[France,[10211,41,100, ...],[10211,41,100, ...], ...], ...]
2017	2	[[Australia,[10223,37,100, ...],[10223,47,100, ...], ...], ...]
2017	3	[[Australia,[10227,25,100, ...],[10227,31,48.52, ...], ...], ...]
2017	4	[[Canada,[10235,24,76.03, ...],[10235,23,96.29, ...], ...], ...]
2017	5	[[Finland,[10247,44,100, ...],[10247,25,100, ...], ...], ...]

✦ 16.3 Import multi-layer JSON file with different-structure detailed data



```
=B1.news(SALES;YEAR,MONTH,COUNTRY,SALES)
```

Use the news() function to combine the year / month fields with the country and monthly sales details.

YEAR	MONTH	COUNTRY	SALES
2017	1	France	[[10211,41,100, ...],[10211,41,100, ...], ...]
2017	1	Japan	[[10210,23,100, ...],[10210,34,100, ...], ...]
2017	1	Spain	[[10212,39,100, ...],[10212,33,100, ...], ...]
2017	1	UK	[[10213,38,94.79, ...],[10213,25,83.39, ...], ...]
2017	1	USA	[[10215,35,100, ...],[10209,39,100, ...], ...]

✦ 16.3 Import multi-layer JSON file with different-structure detailed data



B2:Select data of US and Canada using A2.select(Country.contain(COUNTRY)).

A3~A4: Because the detail data may have different structures, we use full field names as parameters to create a table sequence. The values of a field will be set according to the name, and null values will be set by default if a field does not exist. (As the "ADDRESSLINE1"field and " ADDRESSLINE2" field show below) :

YEAR	COUNTRY	ORDERNUMBER	ADDRESSLINE1	ADDRESSLINE2
2017	USA	10353	2440 Pompton St.	
2017	USA	10352	16780 Pompton St.	
2017	USA	10352	16780 Pompton St.	
2018	USA	10369		
2018	USA	10362		
2018	USA	10371		

✦ 16.3 Import multi-layer JSON file with different-structure detailed data



The final result:

YEAR	COUNTRY	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER
2017	USA	10215	35	100	3
2017	USA	10209	39	100	8
2017	USA	10215	46	100	2
2017	USA	10215	27	89.38	10
2017	USA	10215	33	43.13	9
2017	USA	10215	49	100	4

Now a multi-layer JSON file with different detailed data structure is transformed into a two-dimensional table.

✦ 16.4 Nested aggregation



Aggregate a set of sequences in a JSON file to get SUM.

Calculate the total sales amount in 2016 based on the sales data in JSON format.

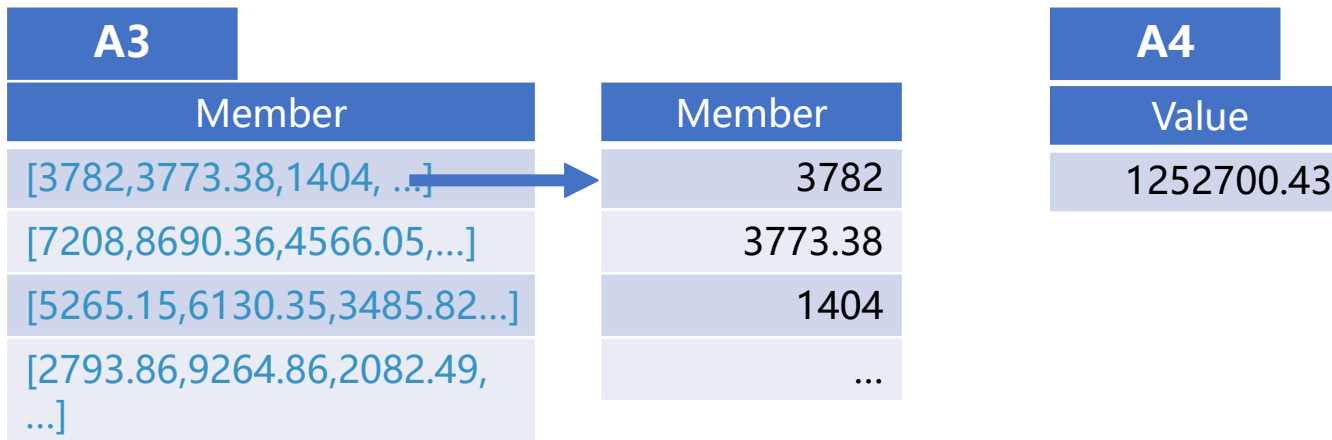
```
[
  {YEAR:2016,MONTH:1,SALES:[
    {ORDERNUMBER:10101, ORDERLINENUMBER:4, SALES:3782, ...},
    {ORDERNUMBER:10102, ORDERLINENUMBER:1, SALES:3773.38, ...},
    ...]
  },
  {YEAR:2016,MONTH:2,SALES:[
    {ORDERNUMBER:10105, ORDERLINENUMBER:2, SALES:7208 ...},
    {ORDERNUMBER:10106, ORDERLINENUMBER:15, SALES:8690.36, ...},
    ...]
  },
  ...]
```

✦ 16.4 Nested aggregation



SPL script is as follows, where A.() is used do loop calculation:

	A	B
1	=json(file("sales.json").read())	/Import JSON data
2	=A1.select(YEAR=2016)	/Select sales data in 2016
3	=A2.field@r("SALES")	/Get sales field values recursively
4	=A3.(~.sum()).sum()	/Use A.() to calculate the subtotal of each group in loop, and then calculate the total sales



✦ 16.5 Get field values recursively & combine members of sequences recursively to get SUM



Get the specified field in the JSON file recursively and combine all the members of the result.
Based on the JSON data of confirmed cases of new coronavirus on a certain day, count the total number of confirmed cases worldwide.

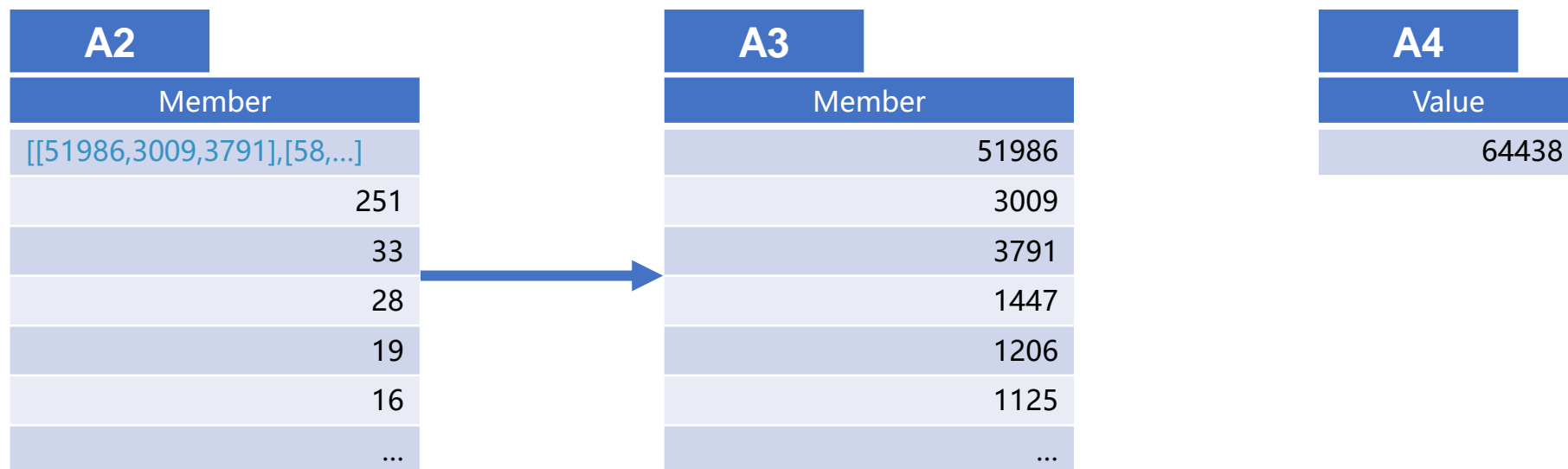
```
[
  {Region:"China",Confirmed:[
    {Region:"Hubei",Confirmed:[
      {Region:"Wuhan",Confirmed:51986},
      {Region:"Xiaogan",Confirmed:3009},
      {Region:"Huanggang",Confirmed:3791},
      ...]
    },
    {Region:"Taiwan",Confirmed:18},
    ...]
  },
  {Region:"Thailand",Confirmed:33},
  ...]
```

✦ 16.5 Get field values recursively & combine members of sequences recursively to get SUM



SPL script is as follows, where A.conj@r() function merges members of sequences recursively:

	A	B
1	=json(file("COVID-19.json").read())	/Import JSON data
2	=A1.field@r("Confirmed")	/Use @r option with A.field() function to get all "Confirmed" field values recursively
3	=A2.conj@r()	/Use @r option with A.conj() function to combine members of sequences recursively
4	=A3.sum()	/Get the sum



✦ 16.6 Store a JSON file to the database



Store the data in a JSON file to the database.

Take the JSON file recording product order information as an example, update the parsed table sequence to the Product table in the database.

JSON file:

```
[{"PRODUCT_ID":1,"PRODUCT_NAME":  
"Apple Juice",  
"SUPPLIER_ID":2,"CATEGORY_ID":1, ...},  
{"PRODUCT_ID":2,"PRODUCT_NAME":  
"Milk",  
"SUPPLIER_ID":1,"CATEGORY_ID":1, ...},  
{"PRODUCT_ID":3,"PRODUCT_NAME":  
"Tomato sauce",  
"SUPPLIER_ID":1,"CATEGORY_ID":2, ...},  
...]
```

Database table:

Product
PRODUCT_ID
PRODUCT_NAME
SUPPLIER_ID
CATEGORY_ID
...



✦ 16.6 Store a JSON file to the database



It's very simple for SPL to import a table sequence into database. Just use `db.update()` function to do that. The SPL script is as follows:

	A
1	<code>=json(file("product.json").read())</code>
2	<code>=connect("db")</code>
3	<code>=A2.update(A1, Product)</code>

A1:Import JSON file as a table sequence.

A2:Connect data source.

A3:Use `db.update()` function to update the table sequence imported in A1 to the Product table in the database. Since the primary key parameter is omitted in the update function, the update will be performed according to the primary key of the database table Product. If the product table has no primary key, update the database according to A1's primary key. If both haven't the primary key, update according to the first field.

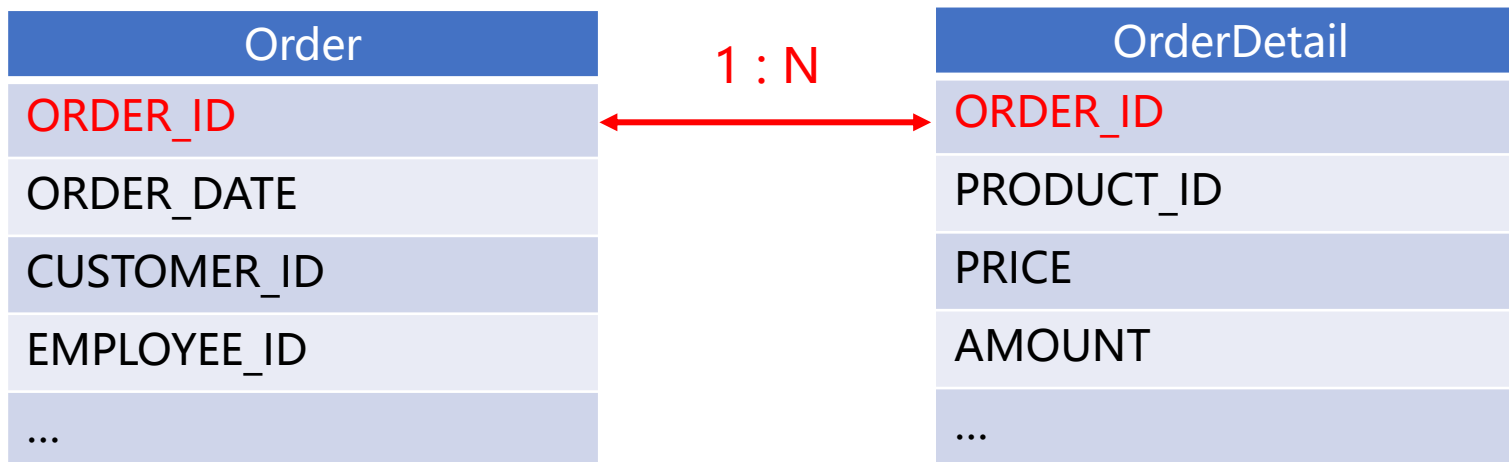
✦ 16.7 Store a multi-layer JSON file to multiple database tables



Import data from a multi-layer JSON file and store it into multiple tables in the database.

Take the JSON file recording the order information as an example. The JSON data has two layers: the first layer is order, and the second layer is order details. Update the orders in and after 2018 and their details to the order table and order details table respectively in the database.

```
[{"ORDER_ID":10248,"ORDER_DATE":"2012-07-04",...,"ORDER_DETAILS":[  
  {"PRODUCT_ID":17,"PRICE":14,"AMOUNT":12, ...},  
  {"PRODUCT_ID":42,"PRICE":9,"AMOUNT":9, ...},  
  ...]}, ...]
```



✦ 16.7 Store a multi-layer JSON file to multiple database tables



The SPL script is as follows:

	A	B
1	=json(file("orders.json").read())	=A1.select(year(ORDER_DATE)>=2018)
2	=connect("demo")	
3	=B1.fname().delete(B1.fname().len())	=A2.update(B1,Order,{A3.concat@c()})
4	=B1.conj(ORDER_DETAILS.derive(B1.ORDER_ID:ORDER_ID))	=A2.update(A4,OrderDetail)

✦ 16.7 Store a multi-layer JSON file to multiple database tables



```
=json(file("orders.json").read())
```

First, import the two-layer JSON file as a two-layer table too:

ORDER_ID	ORDER_DATE	CUMSTOMER_ID	ORDER_DETAILS
10248	2012-07-04	VINET	[[17,14,12,...],[42,9,10,...],...]
10249	2012-07-05	TOMSP	[[14,18,9,...],[51,42,4,...],...]
10250	2012-07-08	HANAR	[[41,7,10,...],[51,42,3,...],...]
10251	2012-07-08	VICTE	[[22,16,6,...],[57,15,15,...],...]
10252	2012-07-09	SUPRD	[[20,64,40,...],[33,2,25,...],...]



PRODUCT_ID	PRICE	AMOUNT	...
20	64	40	...
33	2	25	...
...

✦ 16.7 Store a multi-layer JSON file to multiple database tables



The order date field is in the first layer. We can select the data of and after 2018 directly.

```
=A1.select(year(ORDER_DATE)>=2018)
```

The result is as follows:

ORDER_ID	ORDER_DATE	CUMSTOMER_ID	ORDER_DETAILS
10808	2018-01-01	OLDWO	[[56,38,20,...],[76,18,50,...]]
10809	2018-01-01	WELLI	[[52,7,20,...]]
10810	2018-01-01	LAUGB	[[13,6,7,...],[25,14,5,...],...]

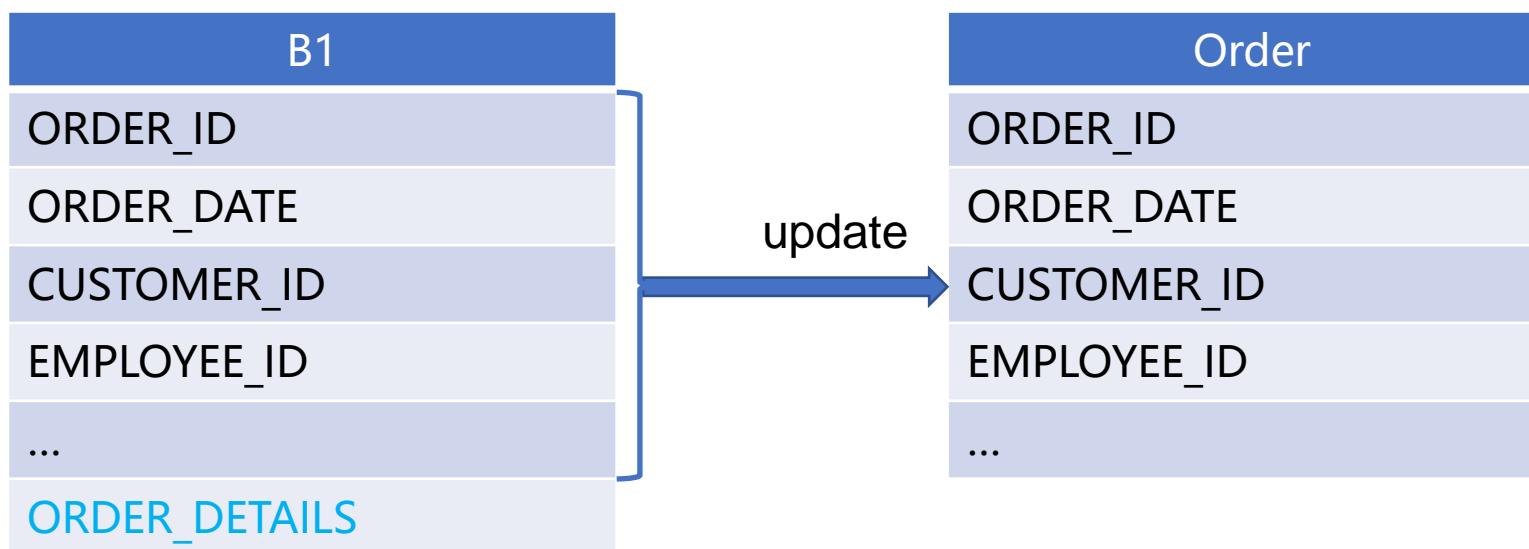
✦ 16.7 Store a multi-layer JSON file to multiple database tables



```
=B1.fname().delete(B1.fname().len())
```

```
=A2.update(B1,Order,{A3.concat@c()})
```

After connecting to the database, first update the main table (order table). However, B1 has one more ORDER_DETAILS field than the database table. Use B1.fname() to get all the field names, and then delete the last member. When using the update function to update, you need to specify the to-be-updated fields. The field parameter uses a macro.

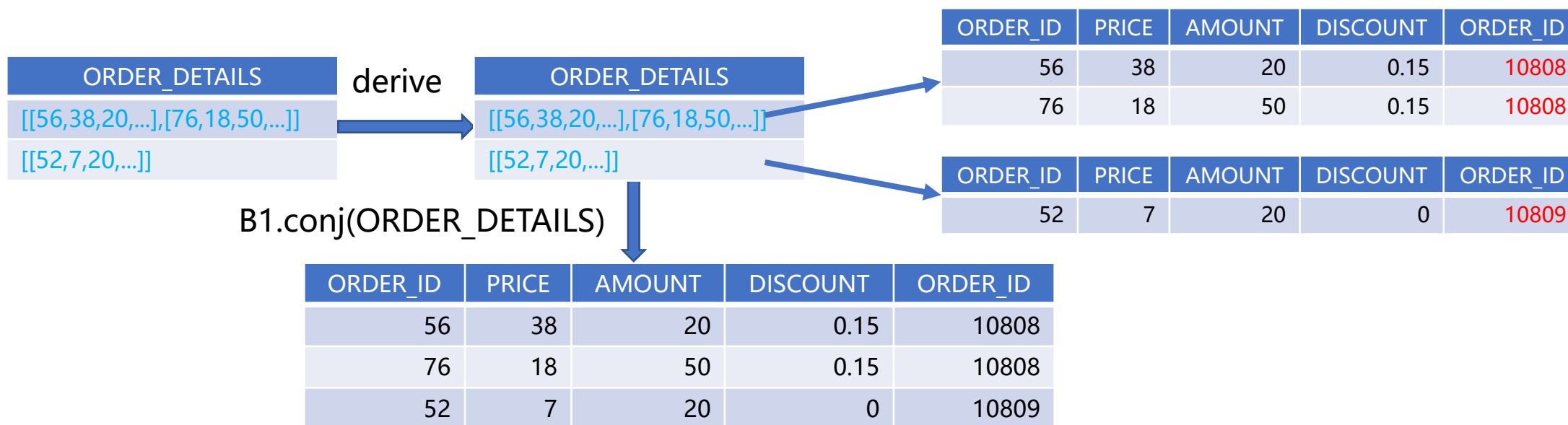


✦ 16.7 Store a multi-layer JSON file to multiple database tables



```
=B1.conj(ORDER_DETAILS.derive(B1.ORDER_ID:ORDER_ID))
```

First, use the `derive()` function to add the order ID field to `ORDER_DETAILS`. Then use `conj()` function to expand the `ORDER_DETAILS` field of each order and put them together. Now we have an order details table with same structure as the database order details table. The results are as follows:

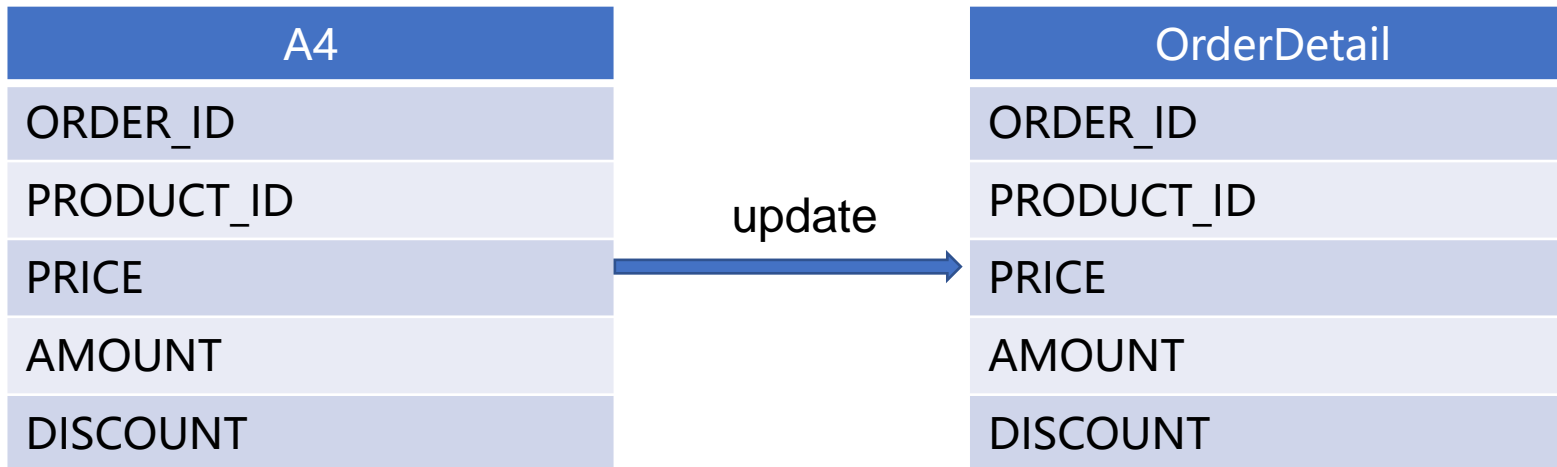


✦ 16.7 Store a multi-layer JSON file to multiple database tables



```
=A2.update(A4,OrderDetail)
```

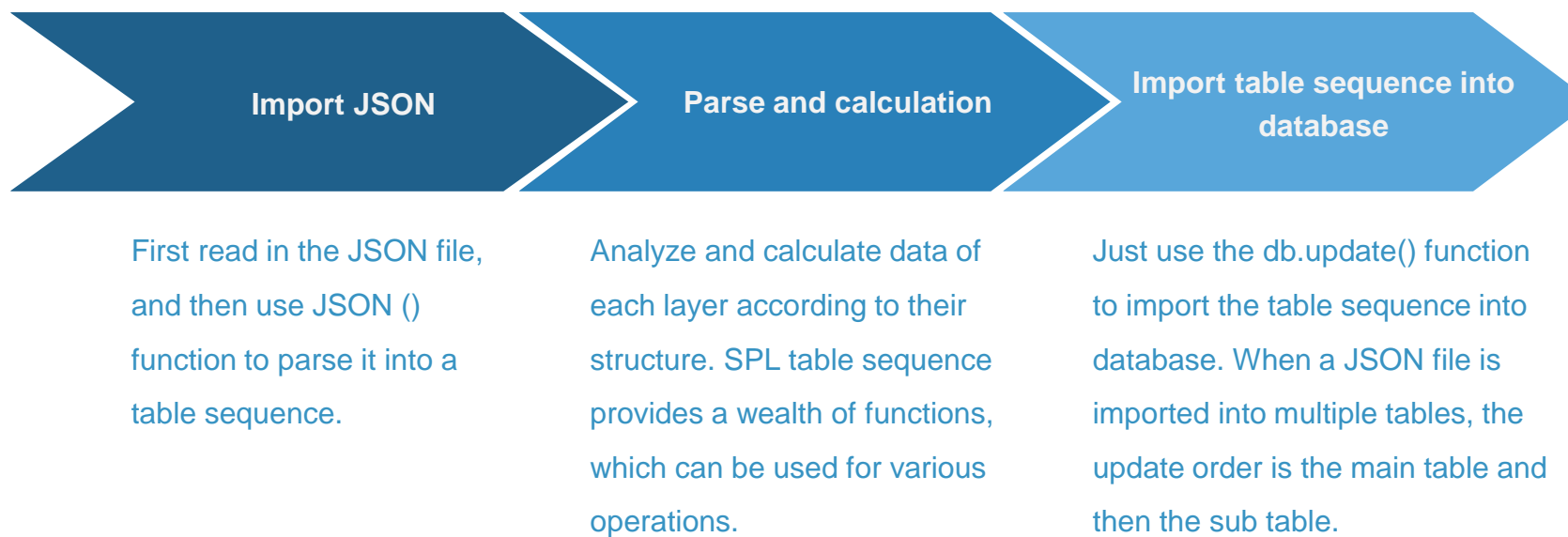
Finally, update the sub table order details. Since the data structure of A4 is consistent with the database table, you don't need to specify the to-be-updated fields.



✦ Summary



JSON data handling workflow



As we can see from the previous sections, the focus of using JSON data is the parsing and calculation. When dealing with multi-layer and complex data with different structures, SPL can simply use "table. field" to reference members, and have rich functions to do calculations.

✦ 16.8 Output the data table as an XML string with only elements



Output the data table as an XML string with only elements.

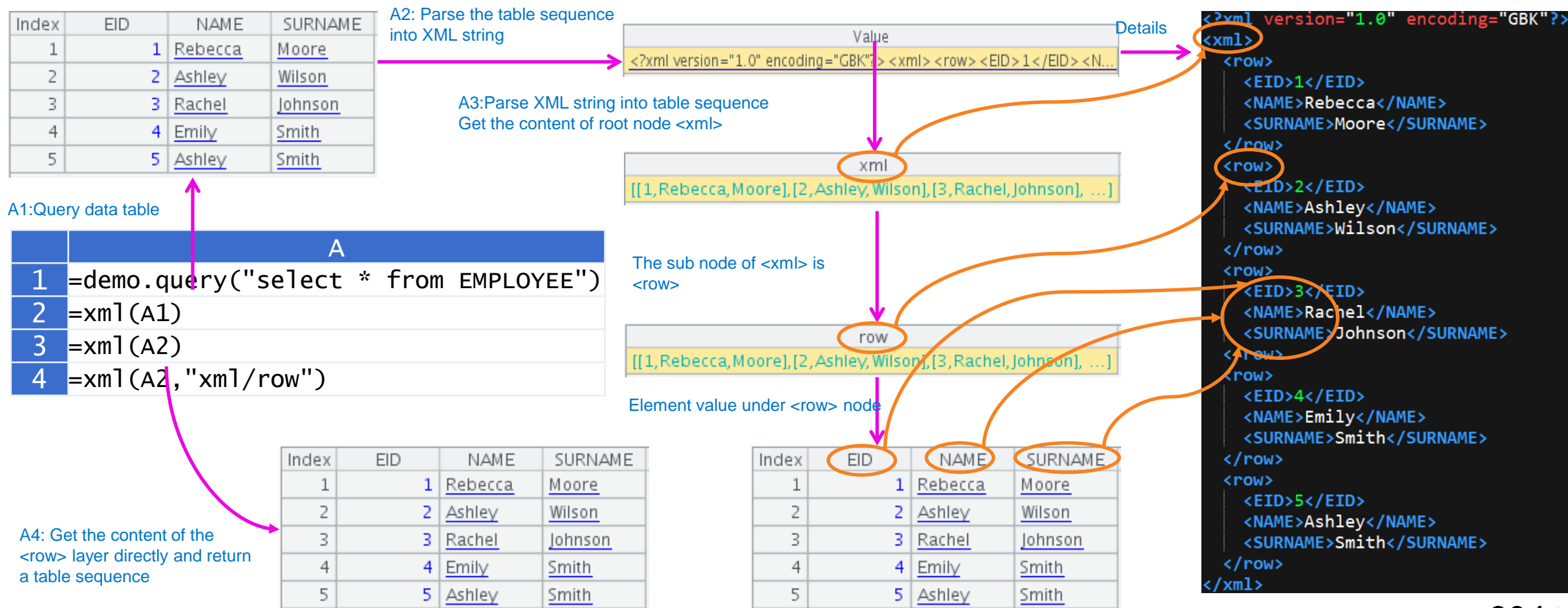
Index	EID	NAME	SURNAME
1	1	Rebecca	Moore
2	2	Ashley	Wilson
3	3	Rachel	Johnson
4	4	Emily	Smith
5	5	Ashley	Smith

◆ 16.8 Output the data table as an XML string with elements only



The processing method is as follows:

SPL provides XML () function to deal with all kinds of XML data conveniently.



✦ 16.9 Import an element-only XML file and organize it according to specified format



Import an element-only XML file and organize it according to the specified format.

```
<?xml version="1.0" encoding="utf-8"?>
<BookStore>
  <Book>
    <title>History</title>
    <author>Tom</author>
    <copies>10</copies>
    <price>80</price>
  </Book>
  <Book>
    <title>Basic Mathematics</title>
    <author>Roy</author>
    <author>Jon</author>
    <copies>5</copies>
    <price>100</price>
  </Book>
  <Book>
    <title>Java</title>
    <author>Harry</author>
    <author>Potter</author>
    <copies>6</copies>
    <price>100</price>
  </Book>
</BookStore>
```

- Multiple books form the bookstore list
- Each book may have multiple authors that need to be merged as a sequence. There may be illegal characters in copies that need to be parsed as numeric value.

✦ 16.9 Import an element-only XML file and organize it according to specified format



The processing method is as follows:

result				
Index	title	author	copies	price
1	History	Tom	10	80
2	Basic Mathematics	Roy&Jon	5	100
3	Java	Harry&Potter	6	100

Index	title	author	copies	price
1	History	Tom	10;	80
2	Basic Mathematics	[Roy,Jon]	5	100
3	Java	[Harry,Potter]	6	100

Parsed as integer

Use & to concatenate multiple values

A2 : Structured XML

	A	B
1	=file("/workspace/BookStore.xml")	/Open xml file
2	=xml(A1.read(),"BookStore/Book")	/Parse XML strings as records
3	=A2.new(title:title,if(ifa(author),author.concat("&"),author):author,if(ifstring(copies),int(replace(copies,";", "")),copies):copies,price)	/Generate table sequence

```
<?xml version="1.0" encoding="utf-8"?>
<BookStore>
  <Book>
    <title>History</title>
    <author>Tom</author>
    <copies>10;</copies>
    <price>80</price>
  </Book>
  <Book>
    <title>Basic Mathematics</title>
    <author>Roy</author>
    <author>Jon</author>
    <copies>5</copies>
    <price>100</price>
  </Book>
  <Book>
    <title>Java</title>
    <author>Harry</author>
    <author>Potter</author>
    <copies>6</copies>
    <price>100</price>
  </Book>
</BookStore>
```

✦ 16.10 Import XML file with both elements and attributes



Import an XML file with both elements and attributes and organize it to structured data.

```
<?xml version="1.0" encoding="utf-8"?>
<bookstore>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author country="it">J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author country="uk">Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

strings like `<K F=v F=v ...>D</K>` into records with K,F,... as fields, the value of K is D. When D has multiple layers of contents, it's parsed as record sequence. In case of `<K .../>`, D is parsed as null. In the case of `<K...></K>`, D is parsed as empty string.

◆ 16.10 Import XML file with both elements and attributes



The processing method is as follows:

Index	bookstore
1	[[[Harry Potter,en],[J K. Rowling,it],[2005],...],[[Learning XML,en],[Eri...

The sub node is <book>, attribute:<category>

Index	book	category
1	[[Harry Potter,en],[J K. Rowli...	CHILDREN
2	[[Learning XML,en],[Erik T. ...	WEB

Elements and attributes under the < book > node

Index	Member
1	[Harry Potter,en]
2	[J K. Rowling,it]
3	[2005]
4	[29.99]

Index	category	title	lang	author	country	year	price
1	CHILDREN	Harry Potter	en	J K. Rowling	it	2005	29.99
2	WEB	Learning XML	en	Erik T. Ray	uk	2003	39.95

A4: Generate a new table sequence and get the corresponding element values and attribute values

A2: Parse into multi-level sequence

```
<?xml version="1.0" encoding="utf-8"?>
<bookstore>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author country="it">J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author country="uk">Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

A1: Read XML strings from file

Value
<?xml version="1.0" encoding="utf-8"?> <bookstore> <book category="...

	A
1	=file("/root/workspace/book.xml").read()
2	=xml@s(A1)
3	=xml@s(A1).bookstore
4	=A3.new(category,book(1).title:title,...)



✦ 16.11 Import XML file, perform alignment merging and then filtering



Import XML data with both elements and attributes, merge records in alignment and then perform filtering

```
<?xml version="1.0"?>
<library>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author name="Giada De Laurentiis" country="it"/>
    <year>2005</year>
    <info>Hello Italian!</info>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author name="J K. Rowling" country="uk"/>
    <year>2006</year>
    <info>Hello Potter!</info>
  </book>
  <book category="WEB">
    <title lang="en">XQuery Kick Start</title>
    <author name="James McGovern" country="us"/>
    <author name="Per Bothner" country="us"/>
    <year>2005</year>
    <info>Hello XQuery!</info>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author name="Erik T. Ray" country="us"/>
    <year>2003</year>
    <info>Hello XML!</info>
  </book>
</library>
```

- Each book may have multiple authors whose a name attribute and country attribute need to be combined in alignment into one column
- You can perform filtering and grouping after the structuralization

◆ 16.11 Import XML file, perform alignment combining and then filtering



The processing method is as follows:

1. Structuralize XML, in which author is list, and the values of name attribute and country attribute need to be respectively concatenated with commas

Index	category	year	title	lang	info	name	country
1	COOKING	2005	Everyday Italian	en	Hello Italian!	Giada De Laurentiis	it
2	CHILDREN	2006	Harry Potter	en	Hello Potter!	J K. Rowling	uk
3	WEB	2005	XQuery Kick Start	en	Hello XQue.	James McGovern,Per Bothner	us,us
4	WEB	2003	Learning XML	en	Hello XML!	Erik T. Ray	us

result1

2. Structuralize XML, in which the author is list, and the values of name attribute and country attribute need to be combined in alignment into one column

Index	title	category	year	author	info
1	Everyday Italian	COOKING	2005	Giada De Laurentiis[it]	Hello Italian!
2	Harry Potter	CHILDREN	2006	J K. Rowling[uk]	Hello Potter!
3	XQuery Kick Start	WEB	2005	James McGovern[us],Per Bothner[us]	Hello XQuery!
4	Learning XML	WEB	2003	Erik T. Ray[us]	Hello XML!

result2

3. On the basis of 2, select the book information in 2005 only

Index	title	category	year	author	info
1	Everyday Italian	COOKING	2005	Giada De Laurentiis[it]	Hello Italian!
2	XQuery Kick Start	WEB	2005	James McGovern[us],Per Bothner[us]	Hello XQuery!

result3

```
<?xml version="1.0"?>
<library>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author name="Giada De Laurentiis" country="it"/>
    <year>2005</year>
    <info>Hello Italian!</info>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author name="J K. Rowling" country="uk"/>
    <year>2006</year>
    <info>Hello Potter!</info>
  </book>
  <book category="WEB">
    <title lang="en">XQuery Kick Start</title>
    <author name="James McGovern" country="us"/>
    <author name="Per Bothner" country="us"/>
    <year>2005</year>
    <info>Hello XQuery!</info>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author name="Erik T. Ray" country="us"/>
    <year>2003</year>
    <info>Hello XML!</info>
  </book>
</library>
```

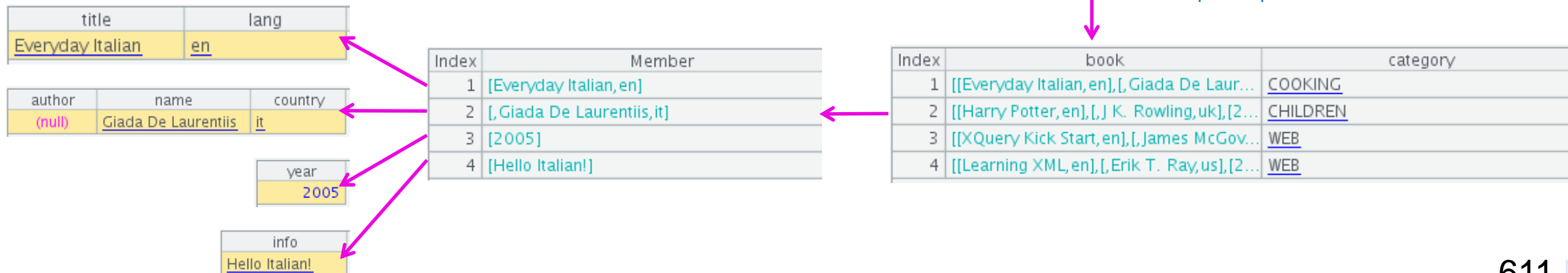

◆ 16.11 Import XML file, perform alignment combining and then filtering



SPL can directly parse and calculate XML. Its agile syntax system needs only several lines of code to get the requirements done.

	A	B
1	<code>=file("/workspace/book1.xml")</code>	/Open the XML file
2	<code>=xml@s(A1.read(),"library/book").library</code>	/Import XML data and obtain the node value
3	<code>=A2.new(category,book.field("year").ifn():year,book.field("title").ifn():title,book.field("lang").ifn():lang,book.field("info").ifn():info,book.field("name").select(~).concat@c():name,book.field("country").select(~).concat(","):country)</code>	/Generate a new table sequence, obtain the values of each field in the sequence while checking whether it is empty; A list value needs to be concatenated as a string
4	<code>=A3.new(title,category,year,(lang,name.array().(~+"")+country.array().(~+"")).concat@c():author,info)</code>	/Generate a new table sequence, where the list column needs to be added in alignment and then concatenated as a string
5	<code>=A4.select(year==2005)</code>	/Perform filtering according to the condition on the basis of A4

A2:Decomposed process



✦ 16.12 Import elements with different structure of the specified layer from an XML file



Import the elements of the specified layer from an XML file with different element structure.

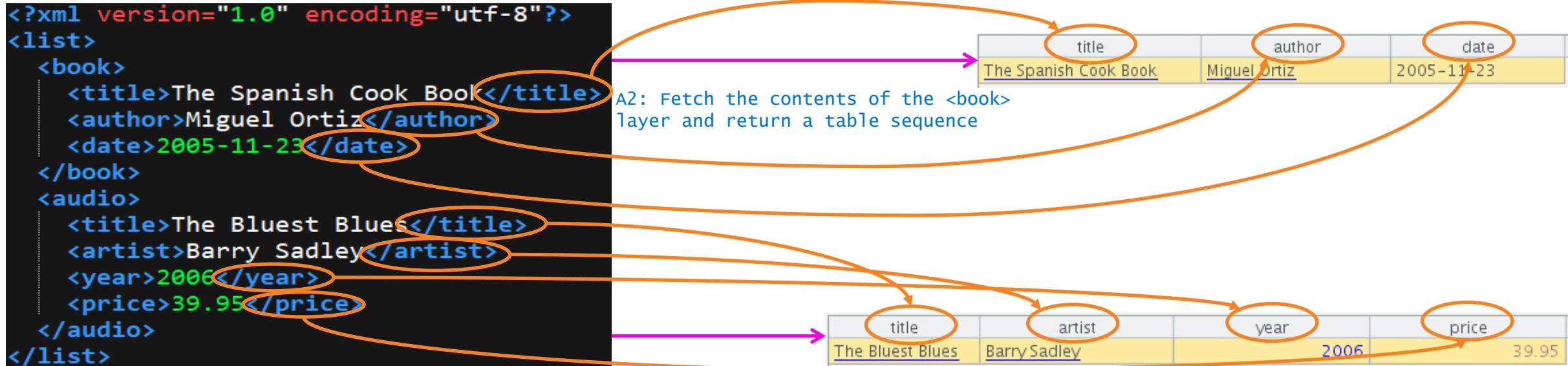
```
<?xml version="1.0" encoding="utf-8"?>
<list>
  <book>
    <title>The Spanish Cook Book</title>
    <author>Miguel Ortiz</author>
    <date>2005-11-23</date>
  </book>
  <audio>
    <title>The Bluest Blues</title>
    <artist>Barry Sadley</artist>
    <year>2006</year>
    <price>39.95</price>
  </audio>
</list>
```

XML (x, s) function, where s represents the layer ID to be fetched, layers are separated by /, and empty indicates fetching from the root. When there are elements of different structures under the node, s can be used to accurately get elements of a certain layer.

✦ 16.12 Import elements with different structure of the specified layer from an XML file



The processing method is as follows:



	A
1	=file("/root/workspace/book.xml").read()
2	=xml(A1,"list/book")
3	=xml(A1,"list/audio")

✦ 16.13 Import elements of the specified layer from an XML file with different sub-node element structure



Import elements of the specified layer from an XML file with different sub-node element structure.

```
<?xml version="1.0" encoding="utf-8"?>
<list>
  <item>
    <table1>
      <row>
        <column1>item 1 table 1 row 1 col 1</column1>
        <column2>item 1 table 1 row 1 col 2</column2>
      </row>
      <row>
        <column1>item 1 table 1 row 2 col 1</column1>
        <column2>item 1 table 1 row 2 col 2</column2>
      </row>
    </table1>
    <table2>
      <row>
        <columnX>item 1 table 2 row 1 col 1</columnX>
        <columnY>item 1 table 2 row 1 col 2</columnY>
        <columnZ>item 1 table 2 row 1 col 3</columnZ>
      </row>
      <row>
        <columnX>item 1 table 2 row 2 col 1</columnX>
        <columnY>item 1 table 2 row 2 col 2</columnY>
        <columnZ>item 1 table 2 row 2 col 3</columnZ>
      </row>
    </table2>
  </item>
  <item>
  </item>
</list>
```

- List of multiple items
- Each item has a fixed number of tables, and the tables are different. Each table has a variable number of rows

✦ 16.13 Import elements of the specified layer from an XML file with different sub-node element structure



The processing method is as follows:

ItemID	column1	column2
1	item 1 table 1 row 1 col 1	item 1 table 1 row 1 col 2
1	item 1 table 1 row 2 col 1	item 1 table 1 row 2 col 2
2	item 2 table 1 row 1 col 1	item 2 table 1 row 1 col 2
2	item 2 table 1 row 2 col 1	item 2 table 1 row 2 col 2

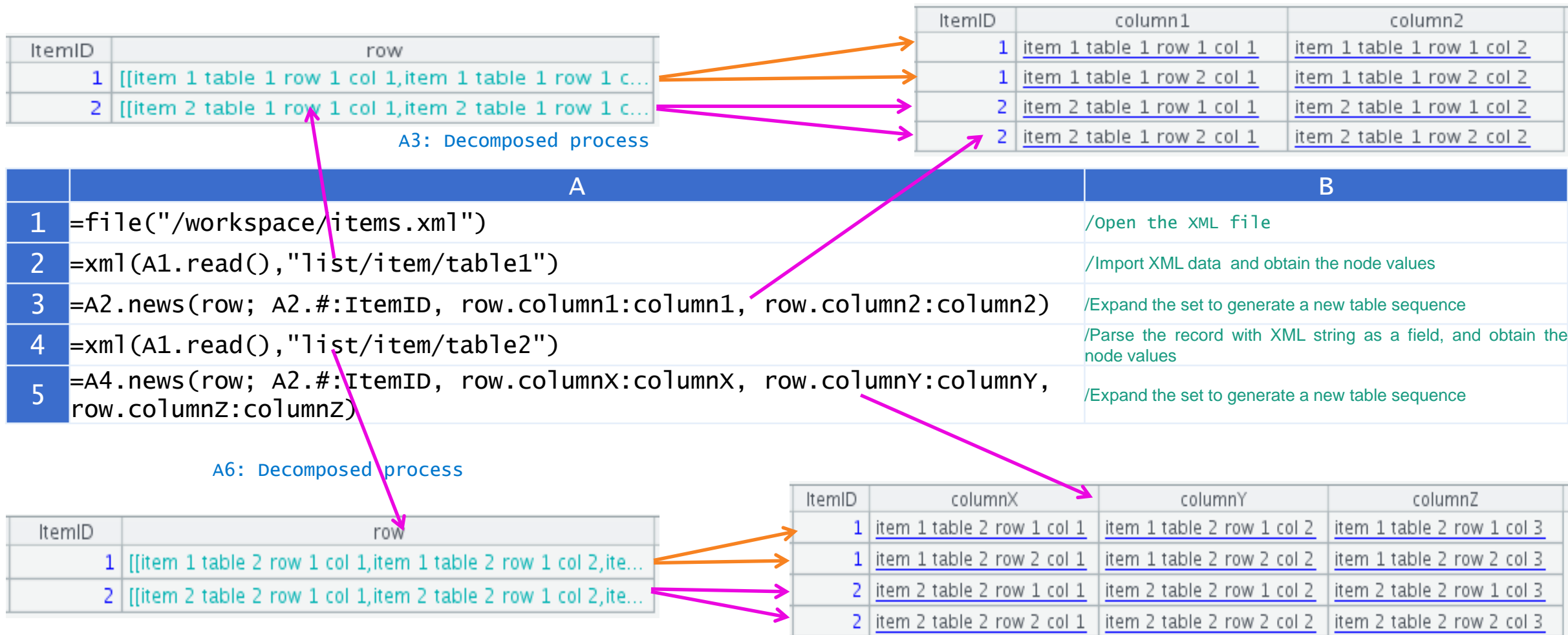
ItemID	columnX	columnY	columnZ
1	item 1 table 2 row 1 col 1	item 1 table 2 row 1 col 2	item 1 table 2 row 1 col 3
1	item 1 table 2 row 2 col 1	item 1 table 2 row 2 col 2	item 1 table 2 row 2 col 3
2	item 2 table 2 row 1 col 1	item 2 table 2 row 1 col 2	item 2 table 2 row 1 col 3
2	item 2 table 2 row 2 col 1	item 2 table 2 row 2 col 2	item 2 table 2 row 2 col 3

```
<?xml version="1.0" encoding="utf-8"?>
<list>
  <item>
    <table1>
      <row>
        <column1>item 1 table 1 row 1 col 1</column1>
        <column2>item 1 table 1 row 1 col 2</column2>
      </row>
      <row>
        <column1>item 1 table 1 row 2 col 1</column1>
        <column2>item 1 table 1 row 2 col 2</column2>
      </row>
    </table1>
    <table2>
      <row>
        <columnX>item 1 table 2 row 1 col 1</columnX>
        <columnY>item 1 table 2 row 1 col 2</columnY>
        <columnZ>item 1 table 2 row 1 col 3</columnZ>
      </row>
      <row>
        <columnX>item 1 table 2 row 2 col 1</columnX>
        <columnY>item 1 table 2 row 2 col 2</columnY>
        <columnZ>item 1 table 2 row 2 col 3</columnZ>
      </row>
    </table2>
  </item>
  <item>
  </item>
</list>
```

✦ 16.13 Import elements of the specified layer from an XML file with different sub-node element structure



Specify the level ID in the function to obtain the element values of this level accurately.



✦ 16.14 Join query over XML file and database data



Perform a join query over an XML file and a database table.

cities				
Index	CID	NAME	POPULATION	STATEID
1	1	New York	8084316.0	2
2	2	Los Angeles	3798981.0	5
3	3	Chicago	2886251.0	1
4	4	Houston	2009834.0	1
5	5	Philadelphia	1492231.0	2
6	6	Phoenix	1371960.0	1

```
state.xml
<?xml version="1.0" encoding="utf-8"?>
<data>
  <state>
    <STATEID>1</STATEID>
    <NAME>"Alabama"</NAME>
    <ABBR>"AL"</ABBR>
  </state>
  <state>
    <STATEID>2</STATEID>
    <NAME>"Alaska"</NAME>
    <ABBR>"AK"</ABBR>
  </state>
</data>
```

The cities table is from MySQL database, and the state data is from XML file. Join them then group and aggregate to count the population of each state.

✦ 16.14 Join query over XML file and database data



We want to get the following result after the data is joined.

result		
Index	STATE	POPULATION
1	Alabama	6268045.0
2	Alaska	9576547.0

JOIN

cities				
Index	CID	NAME	POPULATION	STATEID
1	1	New York	8084316.0	2
2	2	Los Angeles	3798981.0	5
3	3	Chicago	2886251.0	1
4	4	Houston	2009834.0	1
5	5	Philadelphia	1492231.0	2
6	6	Phoenix	1371960.0	1

state.xml

```
<?xml version="1.0" encoding="utf-8"?>
<data>
  <state>
    <STATEID>1</STATEID>
    <NAME>"Alabama"</NAME>
    <ABBR>"AL"</ABBR>
  </state>
  <state>
    <STATEID>2</STATEID>
    <NAME>"Alaska"</NAME>
    <ABBR>"AK"</ABBR>
  </state>
</data>
```

✦ 16.14 Join query over XML file and database data



SPL can directly read XML and MySQL data for mixed calculation; it provides a consistent calculation interface, and various data sources can be calculated in a unified style.

	A	B
1	<code>=mysql.query("select * from cities where STATEID<=2")</code>	<code>/Query cities table</code>
2	<code>=xml(file("/workspace/state.xml").read(),"data/state")</code>	<code>/Parse the record with XML string as a field, and obtain the node value</code>
3	<code>=A2.new(STATEID,NAME,ABBR).keys(STATEID)</code>	<code>/Generate a table sequence and set primary key</code>
4	<code>>A1.switch(STATEID,A3:STATEID)</code>	<code>/Switch STATEID to corresponding records</code>
5	<code>=A1.groups(STATEID.NAME:STATE;sum(POPULATION):POPULATION)</code>	<code>/Group and aggregate</code>

A3:Structuralized XML data

Index	CID	NAME	POPULATION	STATEID
1	1	New York	8084316.0	2
2	3	Chicago	2886251.0	1
3	4	Houston	2009834.0	1
4	5	Philadelphia	1492231.0	2
5	6	Phoenix	1371960.0	1

Index	STATEID	NAME	ABBR
1	1	Alabama	AL
2	2	Alaska	AK

Executed A4

Index	CID	NAME	POPULATION	STATEID
1	1	New York	8084316.0	2
2	3	Chicago	2886251.0	1
3	4	Houston	2009834.0	1
4	5	Philadelphia	1492231.0	2
5	6	Phoenix	1371960.0	1

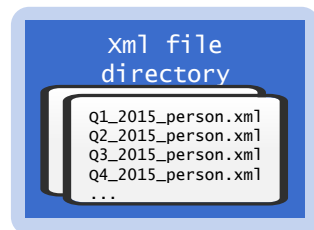
STATEID	NAME	ABBR
2	Alaska	AK

STATEID	NAME	ABBR
1	Alabama	AL

✦ 16.15 Parse XML data in batches



Parse all XML files in the specified directory in batches.



person.xml

```
<?xml version="1.0"?>
<Overlay>
  <Overlay_Filename>arrow_1.png</Overlay_Filename>
  <Overlay_Name>Arrow</Overlay_Name>
  <Overlay_XGrow>100</Overlay_XGrow>
  <Overlay_YGrow>100</Overlay_YGrow>
  <Text_Top>40</Text_Top>
  <Text_Left>10</Text_Left>
  <Text_Bottom>40</Text_Bottom>
  <Text_Right>25</Text_Right>
  <Vector_Grow>0</Vector_Grow>
  <CanRotate_90>1</CanRotate_90>
  <CanRotate_180>1</CanRotate_180>
  <CanRotate_270>1</CanRotate_270>
  <CanFlip_X>0</CanFlip_X>
  <CanFlip_X90>0</CanFlip_X90>
  <CanFlip_X180>0</CanFlip_X180>
  <CanFlip_X270>0</CanFlip_X270>
  <Fill_Color>1</Fill_Color>
  <Border_Color>1</Border_Color>
</Overlay>
```

- There are multiple XML files in the directory, and each XML has the same structure
- Batch parsing and structuring

result

Index	Overlay_Filen...	Overlay_Name	Overlay_X...	Overlay_Y...	Text_Top	Text_Left	Text_Bott...	Text_Right	Vector_Gr...	CanRotate...	CanRotate...	CanRotate...	CanFlip_X	CanFlip_X...	CanFlip_X...
1	arrow_1.png	Arrow	100	100	40	10	40	25	0	1	1	1	0	0	0
2	arrow_2.png	2-Sided Arrow	100	100	45	25	45	25	0	1	0	0	0	0	0

✦ 16.15 Parse XML data in batches



The processing method is as follows:

result														
Index	Overlay_File...	Overlay_Name	Overlay_X...	Overlay_Y...	Text_Top	Text_Left	Text_Bott...	Text_Right	Vector_Gr...	CanRotate...	CanRotate...	CanRotate...	CanFlip_X	CanFlip_X...
1	arrow_1.png	Arrow	100	100	40	10	40	25	0	1	1	1	0	0
2	arrow_2.png	2-Sided Arrow	100	100	45	25	45	25	0	1	0	0	0	0

Index	Overlay
1	[arrow_1.png,Arrow,100, ...]
2	[arrow_2.png,2-Sided Arrow,100, ...]

Index	Member
1	/root/workspace/xml/sample/n1.xml
2	/root/workspace/xml/sample/n2.xml

A2:Import each XML file in loop

A1:XML files in the specified directory

	A	B
1	<code>=directory@p("/workspace/tmp/*.xml")</code>	<code>/List file names that match the wildcard path</code>
2	<code>=A1.(xml(file(~).read()))</code>	<code>/Parse each XML string as a record</code>
3	<code>=A2.conj(~.array())</code>	<code>/Merge each sequence</code>

◆ 16.16 Call external Web Service according to parameters and import XML data

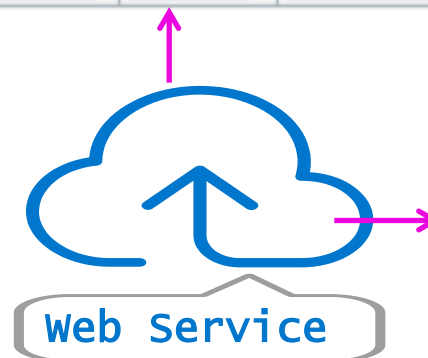


Call external Web Service according to parameters to get XML file data and structuralize the result set.

- Call the external Web Service according to the passed in parameters to return the weather conditions of the region
- Structuralize the XML result set

									result
Index	str1	str2	str3	str4	str5	str6	str7	str8	str9
1	Henan (Province).	Xinyang(City).	464000	57297.jpg	2019/12/20 16:00:31	0°C/10°C	December 20 is cloudy to overcast	East to north is less than category 3

"http://www.webxml.com.cn/Webservices/WeatherWebService.asmx/getWeatherbyCityName?theCityName=%E4%BF%A1%E9%98%B3": "UTF-8"



weather.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfString xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ArrayOfString">
  <string>Henan (Province).</string>
  <string>Xinyang(City).</string>
  <string>464000</string>
  <string>57297.jpg</string>
  <string>2019/12/20 16:00:31</string>
  <string>0°C/10°C</string>
  <string>December 20 is cloudy to overcast</string>
  <string>East to north is less than category 3</string>
  <string>.....</string>
</ArrayOfString>
```

✦ 16.16 Call external Webservice according to parameters and import XML data



The processing method is as follows:

```
"http://www.webxml.com.cn/webServices/WeatherWebService.asmx/getWeatherbyCityName?theCityName=%E4%BF%A1%E9%98%B3":"UTF-8"
```

argCity=Xinyang , A1: Concatenated url

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfString xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <string>Henan (Province).</string>
  <string>Xinyang(City).</string>
  <string>464000</string>
  <string>57297.jpg</string>
  <string>2019/12/20 16:00:31</string>
  <string>0°C/10°C</string>
  <string>December 20 is cloudy to overcast</string>
  <string>East to north is less than category 3</string>
  <string>.....</string> List...
</ArrayOfString>
```

	A	B
1	=wsdl=concat("\http://www.webxml.com.cn/webServices/WeatherWebService.asmx/getWeatherbyCityName?theCityName=",urlencode(argCity,"UTF-8"),"\":"UTF-8\"")	/Based on passed in parameter argCity, concatenate the complete wsdl url
2	=httpfile(\${wsdl})	/Construct httpfile
3	=xml(file(A2).read(),"ArrayOfString/string")	/Parse XML data
4	=create(\${A1.(concat("str",#)).concat@c()})	/Create an empty table sequence
5	>A4.record(A3)	/Fill records in table sequence

✦ 16.17 Get different data from XML file according to parameters



Get different data from an XML file according to parameters.

big.xml

```
<?xml version="1.0" encoding="utf-8"?>
<library>
  <book category="COOKING">
    <title lang="es">The Spanish Cook Book</title>
    <author name="Miguel Ortiz" country="es"/>
    <year>2005</year>
  </book>
  <book category="CHILDREN">
    <title lang="en">Everyone is Super Special</title>
    <author name="Sally Bush" country="us"/>
    <year>2005</year>
  </book>
  <audio format="CD" category="MUSIC">
    <title lang="en">We All Sing Perty</title>
    <artist name="Mary Rogers" country="us"/>
    <year>2006</year>
  </audio>
  <audio format="CD" category="MUSIC">
    <title lang="en">The Bluest Blues</title>
    <artist name="Barry Sadley" country="us"/>
    <year>2006</year>
  </audio>
</library>
```

- An XML file contains multiple label structures, each of which has the label attributes with same number of columns
- Get corresponding data to create different reports according to different parameters

◆ 16.17 Get different data from XML file according to parameters



The processing method is as follows:

Input argument

Title	Value
arg	book

OK Cancel

arg=book, Extract data labeled book

Index	category	title	lang	name	country	year
1	COOKING	The Spanish Cook Book	es	Miguel Ortiz	es	2005
2	CHILDREN	Everyone is Super Special	en	Sally Bush	us	2005

report1

Input argument

Title	Value
arg	audio

OK Cancel

arg=audio, Extract data labeled audio

Index	category	title	lang	name	country	year
1	MUSIC	We All Sing Perty	en	Mary Rogers	us	2006
2	MUSIC	The Bluest Blues	en	Barry Sadley	us	2006

report2

big.xml

```
<?xml version="1.0" encoding="utf-8"?>
<library>
  <book category="COOKING">
    <title lang="es">The Spanish Cook Book</title>
    <author name="Miguel Ortiz" country="es"/>
    <year>2005</year>
  </book>
  <book category="CHILDREN">
    <title lang="en">Everyone is Super Special</title>
    <author name="Sally Bush" country="us"/>
    <year>2005</year>
  </book>
  <audio format="CD" category="MUSIC">
    <title lang="en">We All Sing Perty</title>
    <artist name="Mary Rogers" country="us"/>
    <year>2006</year>
  </audio>
  <audio format="CD" category="MUSIC">
    <title lang="en">The Bluest Blues</title>
    <artist name="Barry Sadley" country="us"/>
    <year>2006</year>
  </audio>
</library>
```

List...

◆ 16.17 Get different data from XML file according to parameters



After SPL parses an XML file, its agile syntax system can complete the logical judgments with only a few lines of code. And its unique macro mechanism greatly improves the code reusability.

	A	B	C
1	=file("/workspace/big.xml")		/Open the XML file
2	=xml@s(A1.read())	=\${arg}=null	/A2:parse XML data; B2:Dedine the macro variable arg, which is null by default
3	=A2.library		/Get library node value
4	for A3	if(A4.fname(1)==arg)	/A4:Loop the nodes; B4:Judge the first field name according to the parameter
5			=\${arg}=if(\${arg}==null,create(category,\${A4.\${arg}.conj(~.fname()).concat@c()}).record(A4.\${arg}.conj(~.array()).insert(1,A4.category)),\${arg}.record(A4.\${arg}.conj(~.array()).insert(1,A4.category)))
6	=\${arg}=\${arg}.new(category,title,lang,name,country,year)		

A3: Get library node values as sequence

Index	Member
1	[[The Spanish Cook Book,es],[Miguel Ortiz,es],[2005], ...]
2	[[Everyone is Super Special,en],[Sally Bush,us],[2005], ...]
3	[[We All Sing Perty,en],[Mary Rogers,us],[2006], ...]
4	[[The Bluest Blues,en],[Barry Sadley,us],[2006], ...]

C5: When the loop variable is empty for the first time, create the columns contained in the empty table sequence, then insert a record into the empty table sequence, and then insert records until the end.

Index	category	title	lang	author	name	country	year
1	<u>COOKING</u>	<u>The Spanish Cook Book</u>	<u>es</u>	(null)	<u>Miguel Ortiz</u>	<u>es</u>	<u>2005</u>
2	<u>CHILDREN</u>	<u>Everyone is Super Special</u>	<u>en</u>	(null)	<u>Sally Bush</u>	<u>us</u>	<u>2005</u>

A6:Generate a new table sequence , return the general columns required by the report

Index	category	title	lang	name	country	year
1	<u>COOKING</u>	<u>The Spanish Cook Book</u>	<u>es</u>	<u>Miguel Ortiz</u>	<u>es</u>	<u>2005</u>
2	<u>CHILDREN</u>	<u>Everyone is Super Special</u>	<u>en</u>	<u>Sally Bush</u>	<u>us</u>	<u>2005</u>



Chapter 17

SPL COOKBOOK

Unstructured text handling

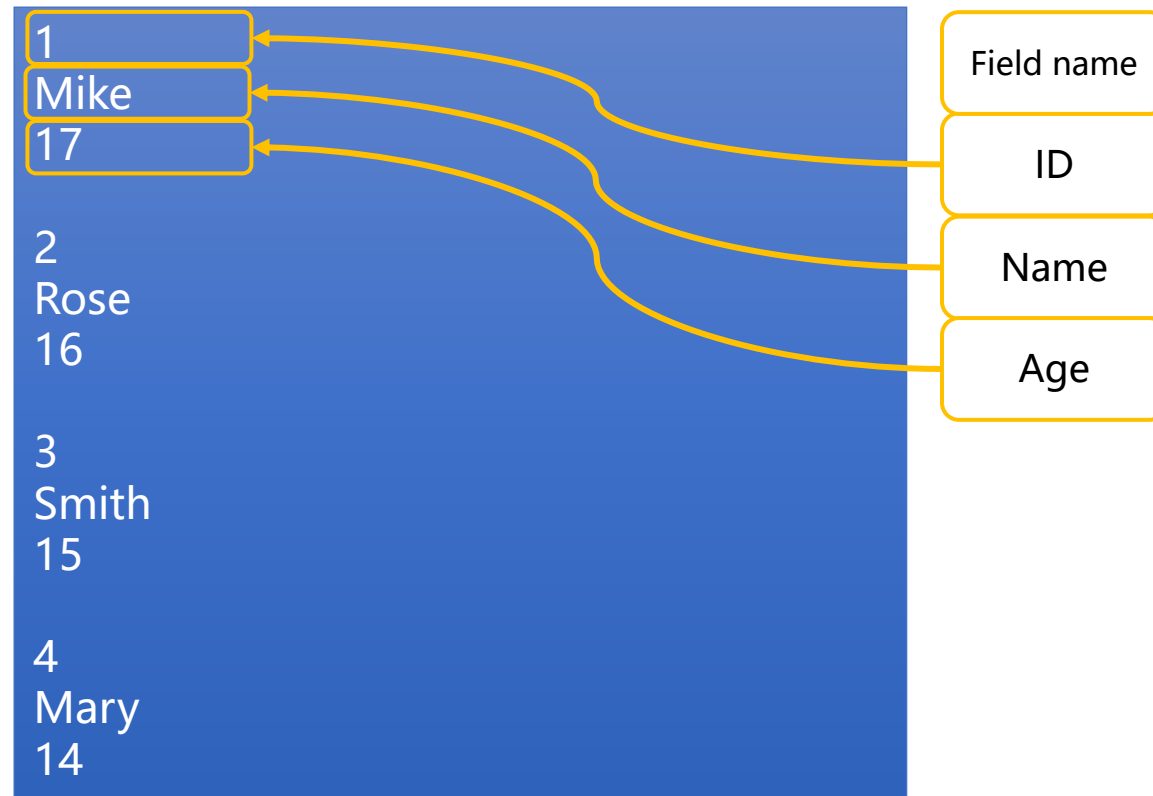
✦ 17.1 Organize a multi-line, fixed-structure text



Organize multi-line fixed-structure records in a text file into structured data.

For example, in *student.txt*, every three lines is a record, and the adjacent records are separated by a blank line.

Read in the file and organize it into structured data.



✦ 17.1 Organize a multi-line, fixed-structure text



For the multi-line text with fixed structure, just read it as a sequence, remove the redundant empty rows, and then insert the members into the target table sequence correspondingly. SPL script is as follows:

	A	B
1	=file("D:\\student.txt")	/Open the file
2	=A1.read@n()	/Read data into sequence by line
3	=A2.select(~!="")	/Remove blank lines between records
4	=create(ID,Name,Age)	/Create table structure
5	=A4.record(A3)	/Populate A3's sequence into the table structure

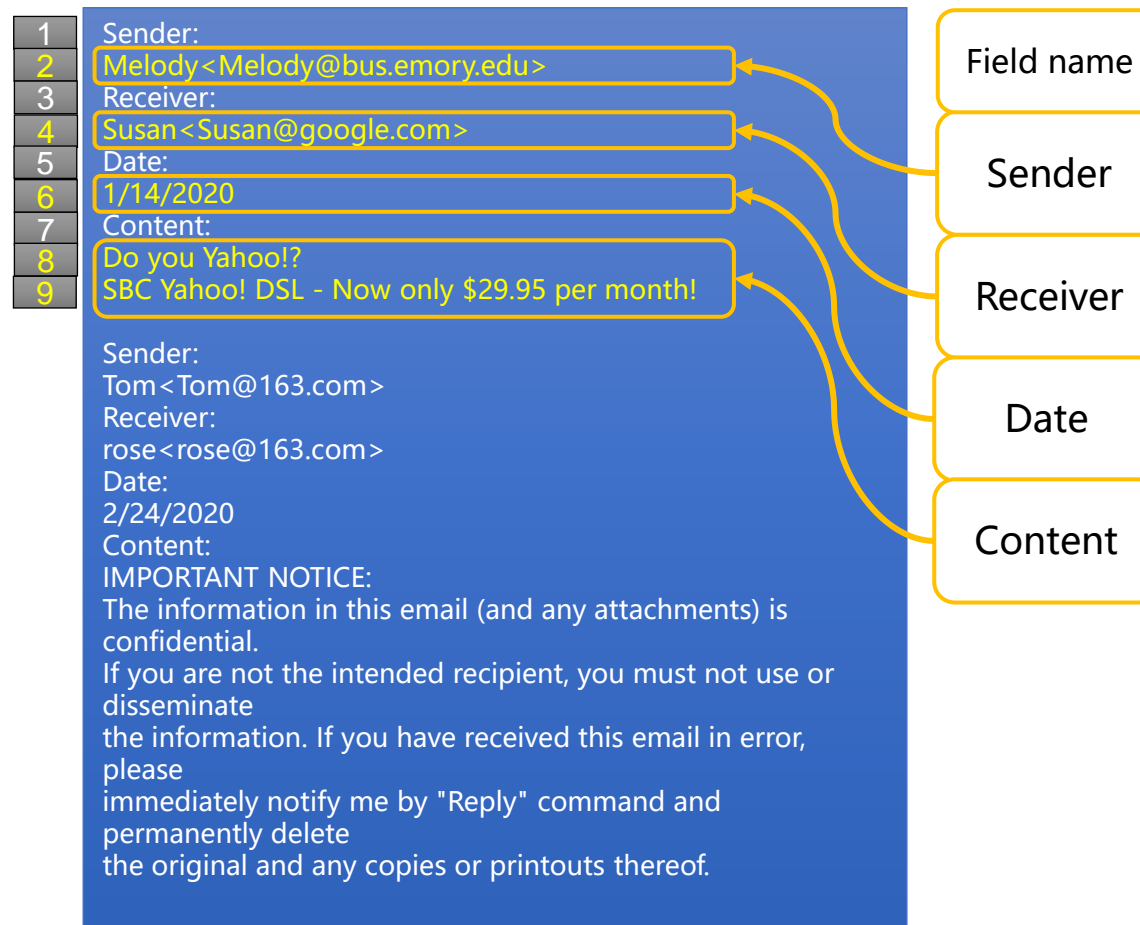
By default, the whole text will be read as a large text string. Here @n option is used to read the text as a sequence line by line, with each line as a member.

✦ 17.2 Organize a varied structure text



Organize texts where a record consists of an indefinite number of lines into structured data. For example, in *mail.txt*, there are sender, receiver, date and mail content. The content may contain multiple lines of information.

The mail's content occupies an indefinite number of lines.



✦ 17.2 Organize a varied structure text



After importing a txt file with a varied structure into a sequence, you need to combine the lines with an indefinite length into one row and then fill values into the table structure correspondingly.

	A	B
1	<code>=file("D:\\mail.txt")</code>	<code>/Open the file</code>
2	<code>=A1.read@n().select(~!="")</code>	<code>/Read it as a sequence and remove blank lines</code>
3	<code>=A2.group@i(~=="Sender:")</code>	<code>/Each line starting with Sender: and its subsequent lines forms a group</code>
4	<code>=A3.new(~(2):Sender,~(4):Receiver,~(6):Date,~.to(8,).concat():Content)</code>	<code>/Extract values of record, merge the body, and create a new table structure</code>
5		

`~.to(8,)` Parameter 8 means starting from line 8 of the current group; the absence of the second parameter means getting all subsequent lines. `.concat()` merges the current sequence of lines into a single row.

✦ 17.3 Parse text with regular expression and organize it into structured data



Parse text with regular expression and organize it into structured data.
For example, organize data in *QQLive.log* into structured data.

The following is the startup log of a video software (QQLive.log).

```
[18-08-13 13:50:21][13104]-  
[0ms][QQLiveMainModule.dll][CQQLiveModule::ParseComman  
dLine] cmd="C:\Program Files  
(x86)\Tencent\QQLive\QQLive.exe"  
[18-08-13 13:50:21][13104]-  
[78ms][QQLiveMainModule.dll]QQLiveDaemon:RegAllHotKey:  
Default Bosskey Registered.  
[18-08-13 13:50:21][13104]-[78ms][QQLiveMainModule.dll]ctrl  
+ alt + shift + 5 Registered  
[18-08-13 13:50:21][13104]-[78ms][QQLiveMainModule.dll]ctrl  
+ alt + shift + 6 Registered  
[18-08-13 13:50:21][13104]-[78ms][QQLiveMainModule.dll]ctrl  
+ alt + shift + 7 Registered
```

✦ 17.3 Parse text with regular expression and organize it into structured data



In the log, each line corresponds to a record. Lines are automatically separated in this example. The yellow part in the figure on the previous page is a single line of data (same structure for the following lines).

It is impossible to use a simple separator to split the lines. Besides, the content contains redundant brackets ([]), minus signs (-), characters (ms), etc.

In this case, a regular expression is used to do the splitting. SPL code is as follows:

	A	B
1	<code>\[(.*)\]\[(.*)\]- \[(.*)ms\]\[(.*)\]\[(.*)\]</code>	/Define a regular expression
2	<code>=file("D:/QQLive.log").read@n()</code>	/Open the log file, and read the contents as a sequence by line
3	<code>=A2.regex(A1)</code>	/Use regex, the regular function for handling sequences, to do the splitting to get fields

✦ 17.4 Parse text with regular expression and organize it into structured data (One record corresponds to multiple lines)



Parse text with regular expression and organize it into structured data. One record is composed of indefinite lines of text.

The following is a software monitoring log (raq.log).

```
[2018-05-14 09:20:20]
SEVERE: Load library module.dll failed.

[2018-05-14 09:20:21]
DEBUG: Temporary file directory is:
D:\temp\esProc\nodes\127.0.0.1_8281\temp.
Files in temporary directory will be deleted on
every 12 hours.

[2018-05-14 09:20:21]
INFO: Server is succeed :started.
```

★ 17.4 Parse text with regular expression and organize it into structured data (One record corresponds to multiple lines)



Here an indefinite number of lines corresponds to one record.

The record boundary needs to be determined first by judging whether a line starts with the left bracket. Then merge members in a group into one string and use regular expression to split it.

SPL script is as follows:

	A	B
1	<code>=file("D:/raq.log").read@n()</code>	/Open the log file, and read the contents into a sequence by line
2	<code>=A1.select(~!="")</code>	/Filter away blank lines
3	<code>=A2.group@i(like(~,"[*"])) .(~.concat(</code> <code>))</code>	/Group by record content and concatenate members in a group into a string
4	<code>\[(.*)\] ([A-Z]+):(.*)</code>	/Define a regular expression
5	<code>=A3.regex(A4)</code>	/Perform regular analysis

✦ 17.5 Read in text and perform transposition



Read in the crosstab data in a text file and then transpose it to structured data.

The following is a score cross table in CSV format (scores.csv).

```
ID,Name,Math,Physics,Chemistry
1,Mike,67,87,72
2,Rose,80,90,84
3,Smith,90,88,76
4,Mary,88,67,77
5,Tod,55,70,87
6,Melody,40,90,55
7,David,90,65,80
8,Snoopy,100,90,85
9,Michale,70,78,55
10,Nikita,66,88,70
```


✦ 17.5 Read in text and perform transposition



The file itself is structured data. It's a cross table of students' scores in each subject. Now you need to rearrange the subjects and scores under the *subject* and *score* fields.

SPL script is as follows:

	A	B
1	<code>=file("D:/scores.csv")</code>	<code>/Open the file</code>
2	<code>=A1.import@tc()</code>	<code>/Import data</code>
3	<code>=A1.pivot@r(ID,Name;Subject,Score)</code>	<code>/Transpose subjects and scores to fields <i>Subject</i> and <i>Score</i></code>
4		

✦ 17.6 Organize a complex text file into structured data



Read data in complex text file and organize it into structured data.

The following is HPUpdate.exe.log for windows.

```
1,"fusion","GAC",0
1,"WinRT","NotApp",1
3,"System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089","C:\windows\assembly\
NativeImages_v4.0.30319_64\System\01a3608d87251d7ea99
342a88d079c23\System.ni.dll",0
3,"System.Core, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089","C:\windows\assembly\
NativeImages_v4.0.30319_64\System.Core\2a6c39230fef9dfaf
c7ede45f99ec776\System.Core.ni.dll",0
3,"WindowsBase, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35","C:\windows\assembly\
NativeImages_v4.0.30319_64\WindowsBase\996cd1a75c20ce
6e697aad199323707b\WindowsBase.ni.dll",0
3,"PresentationCore, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35","C:\windows\assembly\
NativeImages_v4.0.30319_64\PresentationCore\6228d402fde
bfae866e84fdfe08773bf\PresentationCore.ni.dll",0
```

In the log file, each line corresponds to a record. In the figure on the left, lines are distinguished through the change of color. The corresponding 4 columns are named type, desc, file and status. The main points of splitting are as follows:

1: It can't be separated simply by commas. You need to take the pairing of quotes into consideration.

2: Desc contains subfields and is described as a segmented string.

3: Desc of each type is different. There is no subfield such as version when the type is 1.

4: Desc is not a regular segmented string, where the first section is name without section value.

✦ 17.6 Organize a complex text file into structured data



Splitting steps:

First, split the text into a basic table using the comma outside of the quotes and rename the default field:

序号	Type	Desc	File	Status
1	1	fusion	GAC	0
2	1	WinRT	NotApp	1
3	3	System, Version=4.0.0.0, Cult...	C:\windows\las...	0
4	3	System.Core, Version=4.0.0.0...	C:\windows\las...	0
5	3	WindowsBase, Version=4.0.0...	C:\windows\las...	0
6	3	PresentationCore, Version=4...	C:\windows\las...	0
7	3	PresentationFramework, Vers...	C:\windows\las...	0
8	3	System.Xaml, Version=4.0.0.0...	C:\windows\las...	0
9	3	System.Configuration, Versio...	C:\windows\las...	0
10	3	System.Xml, Version=4.0.0.0, ...	C:\windows\las...	0
11	3	System.Web, Version=4.0.0.0...	C:\windows\las...	0
12	3	System.Web.Extensions, Vers...	C:\windows\las...	0
13	3	System.Security, Version=4.0...	C:\windows\las...	0
14	3	PresentationFramework.Aero...	C:\windows\las...	0
15	3	WindowsFormsIntegration, V...	C:\windows\las...	0
16	3	System.Drawing, Version=4.0...	C:\windows\las...	0
17	3	System.Windows.Forms, Ver...	C:\windows\las...	0

Then, split the desc field into a table sequence consisting of name and value key pairs.

序号	name	value
1	Culture	neutral
2	System	
3	PublicKeyToken	b77a5c561934e089
4	Version	4.0.0.0

✦ 17.6 Organize a complex text file into structured data



Splitting steps:

From the table sequence of Desc key value pairs, Extract the key with empty value to the Name column:

序号	Type	Desc	File	Status	Name
1	1	[[fusion,]]	GAC	0	fusion
2	1	[[WinRT,]]	NotApp	1	WinRT
3	3	[[Culture,ne...	C:\windows\assemblyl...	0	System
4	3	[[Culture,ne...	C:\windows\assemblyl...	0	System.Core
5	3	[[Culture,ne...	C:\windows\assemblyl...	0	WindowsB...
6	3	[[Culture,ne...	C:\windows\assemblyl...	0	Presentatio...
7	3	[[Culture,ne...	C:\windows\assemblyl...	0	Presentatio...
8	3	[[Culture,ne...	C:\windows\assemblyl...	0	System.Xaml
9	3	[[Culture,ne...	C:\windows\assemblyl...	0	System.Co...
10	3	[[Culture,ne...	C:\windows\assemblyl...	0	System.Xml
11	3	[[Culture,ne...	C:\windows\assemblyl...	0	System.Web
12	3	[[Culture,ne...	C:\windows\assemblyl...	0	System.We...
13	3	[[Culture,ne...	C:\windows\assemblyl...	0	System.Se...
14	3	[[Culture,ne...	C:\windows\assemblyl...	0	Presentatio...
15	3	[[Culture,ne...	C:\windows\assemblyl...	0	WindowsF...
16	3	[[Culture,ne...	C:\windows\assemblyl...	0	System.Dr...
17	3	[[Culture,ne...	C:\windows\assemblyl...	0	System.Wi...

Then delete the corresponding name line from the key pair table

序号	name	value
1	Culture	neutral
2	System	
3	PublicKeyToken	b77a5c561934e089
4	Version	4.0.0.0

序号	name	value
1	Culture	neutral
2	PublicKeyToken	b77a5c561934e089
3	Version	4.0.0.0

✦ 17.6 Organize a complex text file into structured data



Splitting steps:

Transpose the key pair table to facilitate merging the sub table fields into the main table:

序号	name	value
1	Culture	neutral
2	PublicKeyToken	b77a5c561934e089
3	Version	4.0.0.0

序号	Culture	PublicKeyToken	Version
1	neutral	b77a5c561934e089	4.0.0.0

The transposed sub tables are merged into the main table:

序号	Type	Name	Culture	PublicKeyT...	Version	File	Status
1	1	fusion	(null)	(null)	(null)	GAC	0
2	1	WinRT	(null)	(null)	(null)	NotApp	1
3	3	System	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0
4	3	System.Core	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0
5	3	WindowsBase	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
6	3	PresentationCore	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
7	3	PresentationFra...	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
8	3	System.Xaml	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0
9	3	System.Configur...	neutral	b03f5f7f11...	4.0.0.0	C:\windows\asse...	0
10	3	System.Xml	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0
11	3	System.Web	neutral	b03f5f7f11...	4.0.0.0	C:\windows\asse...	0
12	3	System.Web.Ext...	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
13	3	System.Security	neutral	b03f5f7f11...	4.0.0.0	C:\windows\asse...	0
14	3	PresentationFra...	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
15	3	WindowsFormsIn...	neutral	31bf3856a...	4.0.0.0	C:\windows\asse...	0
16	3	System.Drawing	neutral	b03f5f7f11...	4.0.0.0	C:\windows\asse...	0
17	3	System.Windows...	neutral	b77a5c561...	4.0.0.0	C:\windows\asse...	0

✦ 17.6 Organize a complex text file into structured data



SPL script:

@q option removes the quotation marks of each segmented string; @o means that the quotation mark is an escape character, otherwise the slash sign in the path will be regarded as the default escape character; @c means using comma as the separator.

When merging the sub table to the main table, if a row with type 1 matches no key pair, @1 option is used to left join the Desc key pairs table by the main table.

	A	B
1	<code>=file("D:/HPUpdate.exe.log").import @qoc()</code>	/Open and import the log file
2	<code>=A1.rename(_1:Type,_2:Desc,_3:File,_4:Status)</code>	/Rename the default fields
3	<code>=A2.run(Desc=Desc.property@c())</code>	/Split desc into a key pairs table
4	<code>=A3.derive(Desc.select(value=="").name:Name)</code>	/Extract a Name column from key pairs
5	<code>=A4.run(Desc.delete(Desc.pselect(value=="")))</code>	/Delete the corresponding name in the key pair
6	<code>=A5.run(Desc=Desc.pivot(;name,value))</code>	/Transpose the key pair table
7	<code>=A6.news@1(Desc;Type,Name,Culture,PublicKeyToken,Version,File,Status)</code>	/Merge sub table to main table

✦ 17.7 Search all text files in the specified directory to find the lines containing keywords



Traverse all text files in the specified directory to find the lines containing keywords.

Similar to the `grep` command that searches all text files in the specified directory(including subdirectories) to find the lines containing keywords.

✦ 17.7 Search all text files in the specified directory to find the lines containing keywords



Define two entry parameters. Path is the search root directory and key is the keyword to be searched.

Run is a loop execution function, which traverses and executes all files under the root directory. The second run is to traverse and search the file content.

The logic of output is very simple. After finding it, print the content of the line and line number. In SPL, use / instead of not + to concatenate an integer with a string.

	A	B
1	<code>=directory@ps(path+"/*.txt")</code>	/List all text files in the search directory (including subdirectories)
2	<code>=A1.run(file(~).read@n().run(if(pos(~,key),output(A1.~/No"/#/"Row: "/"~))))</code>	/Read in the contents of each file, compare with the keyword line by line, and output corresponding information

✦ 17.8 Replace string in all text files in a specified directory



Traverse all the files in the specified directory, find and replace the specified text and output the result.

Given the root directory, replace the specified text in all text files under the path.

✦ 17.8 Replace string in all text files in a specified directory



Three entry parameters are defined. path is the root directory, source is the source string to be replaced, and target is the target string to be replaced. SPL script is as follows:

	A	B	C
1	=directory@ps (path+ "/*.txt")	/List all text files in the path directory(including subdirectories)	
2	for A1	=file(A2).read@n()	/Loop through all files in the directory
3		=B2.run(~=replace(~,s ource,target))	/Replace the content in each file
4		=file(A2).write(B3)	/Write out the replaced content to the original file

✦ 17.9 Count the frequencies of each English word in a text file



Count the number of each word's appearances in a text file.

Implement the wordcount algorithm. Given the text file, count the number of each word's frequencies in the text.

✦ 17.9 Count the frequencies of each English word in a text file



An entry parameter needs to be defined. filePath is the target file name with path.

SPL script is as follows:

	A	B
1	<code>=file(filePath).read()</code>	/Read in the content of the given file
2	<code>=A1. words()</code>	/Split the content into a sequence of words
3	<code>=A2.groups(lower(~):Word;count(~):Count)</code>	/Convert all words to lowercase, group them and count their frequencies

✦ 17.10 Remove duplicate lines from a text file



Remove duplicate lines from a text file.

Here are some common network addresses collected by a student (urls.txt) . Sort it out to delete the duplicate URLs.

```
https://123.sogou.com/  
https://www.sogou.com/  
https://stackoverflow.com/  
https://123.sogou.com/  
http://www.raqsoft.com.cn/  
https://www.baidu.com/  
https://www.sogou.com/  
https://123.sogou.com/  
https://stackoverflow.com/  
http://www.raqsoft.com.cn/
```

✦ 17.10 Remove duplicate lines from a text file



Read the the file line by line and group them also by line. Get only the first of the duplicate lines. SPL script is as follows:

	A	B
1	<code>=file("d:/urls.txt"))</code>	/Open the specified file
2	<code>=A1.read@n()</code>	/Read file content by line as sequence
3	<code>=A2.group@1()</code>	/Group by members of the sequence
4	<code>=A1.write(A3)</code>	/Get the deduplicated content

@1 option returns only the first of the duplicate lines to obtain the unique content.

✦ 17.11 Count the frequencies of each letter in a text file



Count the number of appearances of each English letter in a text file.

Implement the wordcount algorithm. Given the text file, count the frequencies of each English letter in the text.

✦ 17.11 Count the frequencies of each letter in a text file



SPL script is as follows:

	A	B
1	<code>=file(filePath).read()</code>	/Read in the content of the given file
2	<code>=A1. split()</code>	/Split the content into a sequence of words
3	<code>=A2.groups(~:Char;count(~):Count)</code>	/Group and count characters

✦ 17.12 Remove duplicate paragraph from a text file



Remove duplicate paragraph from a text file.

Novels copied from online posts (novel.txt) often have duplicate paragraphs.

✦ 17.12 Remove duplicate paragraph from a text file



Deduplication of a novel should not disrupt its original content order. Number the lines with condition `min(#)` and get only the first of the duplicates and records number of the obtained line for restoring the original content order.

Only keep the number of the first-found row among the duplicates

Only the `~` field is exported. All fields will be exported by default.

	A	B
1	<code>=file("d:/novel.txt"))</code>	/Open the specified file
2	<code>=A1.read@n()</code>	/Read the contents by line as a sequence
3	<code>=A2.groups(~;min(#):k)</code>	/Put rows with same content in same group, keeping the smallest row number only
4	<code>=A3.sort(k)</code>	/Sort by row number
5	<code>=A1.export(A4,~)</code>	/Write the sorted content to the original file
6		



Chapter 18

SPL COOKBOOK

String & datetime handling

✦ 18.1 Concatenate strings in two columns



Concatenate strings in two columns into one column.

Get the full name and salary of R&D employees in New York. The employee table is as follows:

ID	NAME	SURNAME	STATE	DEPT	SALARY
1	Rebecca	Moore	California	R&D	7000
2	Ashley	Wilson	New York	Finance	11000
3	Rachel	Johnson	New Mexico	Sales	9000
4	Emily	Smith	Texas	HR	7000
5	Ashley	Smith	Texas	R&D	16000
...

✦ 18.1 Concatenate strings in two columns



SPL script is as follows, in which the operator "+" is used to concatenate strings:

	A	B
1	=connect("db")	/Connect to data source
2	=A1.query("select * from Employee")	/Import employee table
3	=A2.select(STATE=="New York"&&DEPT=="R&D")	/Select employee records of R&D department in New York
4	=A3.new(NAME+" "+SURNAME:FULLNAME, SALARY)	/Use the sign "+" to concatenate strings to form full name

A4	FULLNAME	SALARY
	Matthew Johnson	6000
	Lauren Thomas	12000
	Brooke Williams	12000

✦ 18.2 Concatenate string and other type of value



Concatenate string and other type of value.

Here are two text files. Look for the string of text 1 in text 2 and output a result of the following format:

file1
like parks
went out
go out

file2
I like to go out because I like parks.
Ben does not go out much.
Shelly went out often but does not like parks.
Harry does not go out neither does he like parks.

Output
Q1. like parks
I
Shelly
Harry
Q2. went out
Shelly
Q3. go out
I
Ben
Harry

✦ 18.2 Concatenate string and other type of value



SPL script is as follows, where the symbol '/' is used to concatenate string and other type of value:

	A	B
1	=file("file1.txt").read@n()	/Read text1
2	=file("file2.txt").read@n()	/Read text2
3	=A1.conj(("Q"/#/" " /~) A2.select(pos(~, A1.~)).(~.words()(1)))	/Loop each of the strings in text 1 to find it in text 2, and get the first word. Then precede each group of result with Q and the sequence number of the corresponding string in A1 and its content. The sequence number is an integer, to which another type of value is concatenated using the symbol '/'.

A3	Member
	Q1. like parks
	I
	Shelly
	Harry
	Q2. went out
	Shelly
	Q3. go out
	I
	Ben
	Harry

✦ 18.3 Concatenate members in a sequence



Concatenate members in a sequence into a string.

Table A and table B have same structure. Use table B to update table A. When the primary key of table B exists in table A, update the record; otherwise, add a new row.

Table A		
ID	Amount	...
1	3063.0	...
2	3868.6	...
4	2713.5	...
...

Table B		
ID	Amount	...
1	3063.0	...
2	4507.0	...
3	2713.5	...
...

✦ 18.3 Concatenate members in a sequence



A.concat(d) function concatenates members of a sequence with the separator d and returns a string. @c option indicates that comma is used to connect them. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select COLUMN_NAME from INFORMATION_SCHEMA.KEY_COLUMN_USAGE k where k.TABLE_NAME='B'")	/Get the primary key of table B from the system table. Each database has their own way to get the primary key (Here take MSSQL as an example)
3	=pks=A2.(COLUMN_NAME)	/Define variable pks, which is a sequence of primary key column names
4	=A1.query("select COLUMN_NAME from INFORMATION_SCHEMA.COLUMNS c where c.TABLE_NAME='B'")	/Retrieve all columns of the table
5	=columns=A4.(COLUMN_NAME)	/Define variable columns, which is a sequence of column names
6	= "MERGE INTO A as t USING B as s ON "+pks.("t."+~+"=s."+~).concat(" and ")+" WHEN MATCHED THEN UPDATE SET "+(columns\pks).("t." + ~ + "=s." + ~).concat@c()+ " WHEN NOT MATCHED THEN INSERT VALUES("+columns.("s." + ~).concat@c()+")"	/Dynamically splice a "merge into" statement, where A.concat function is used to concatenate members of a sequence and return a string.
7	=A1.excute(A6)	/Execute the "merge into" statement in A6

✦ 18.4 Add quotation marks to members when concatenating members of a sequence



Add quotation marks to members when concatenating members of a sequence.

Find the states where employees in each department work, and separate state names by spaces.

Since some states have space in their names, they should be distinguished by adding quotation marks to them. The employee table is as follows:

ID	NAME	SURNAME	STATE	DEPT	SALARY
1	Rebecca	Moore	California	R&D	7000
2	Ashley	Wilson	New York	Finance	11000
3	Rachel	Johnson	New Mexico	Sales	9000
4	Emily	Smith	Texas	HR	7000
5	Ashley	Smith	Texas	R&D	16000
...

✦ 18.4 Add quotation marks to members when concatenating members of a sequence



Use @q option with A.concat() function to add double quotation marks to the string member when concatenating members. Similarly, @i option adds single quotation marks to the string member when concatenating members. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Employee")	/Import employee table
3	=A2.group(DEPT; ~.id(STATE):STATES)	/Group by department, get the unique state names in each group
4	=A3.new(DEPT, STATES.concat@q(" "):STATES)	/Concatenate state names in each group into a string using the A.concat() function; @q option adds double quotation marks to string members.

A4	DEPT	STATES
	Administration	"Florida" "Pennsylvania"
	Finance	"California" "Colorado" "Florida" "Georgia" "Illinois" "Michigan" "New Jersey" "New York" "North Carolina"

✦ 18.5 Convert a table sequence to CSV format



Convert a table sequence to CSV format, separate field values in each record with commas.

Convert the *Department* table to CSV format and copy it to the system clipboard. The table is as follows:

ID	Name	Manager
1	Administration	1
2	Finance	4
3	HR	5
4	Marketing	6
5	Production	7
...

✦ 18.5 Convert a table sequence to CSV format



A.export() function concatenates members of as table sequence and returns a string. @t option generates the header line, and @c option generates the CSV format. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Department")	/Import <i>Department</i> table
3	=A2.export@ct()	/Use A.export() function to concatenate members into a string. @c option is used to generate the CSV format, and the @t option is used to generate the header line.
4	=clipboard(A3)	/Copy the string of A3 to the system clipboard

A2	ID	Name	Manager
	1	Administration	1
	2	Finance	4
	3	HR	5

A3	ID,Name,Manager 1,Administration,1 2,Finance,4 3,HR,5 ...
----	---

✦ 18.6 Split a string into a sequence of characters



Split a string into a sequence of characters.

Count the number of commas outside the bracket in the source code of a web page. Part of the source code is as follows:

```
<html>
<b> </b>
<table cellpadding="2.5px" rules="all" style=";background-color: rgb(255,255,255);border: 1px
solid;border-collapse: collapse; border-color: rgb(187,187,187)">
<colgroup><col width="25px" style="background-color: rgb(218,231,245)" /> <col /> <col /> <col
/> <col /> <col /> <col /> </colgroup>
<thead> <tr style=" background-color: rgb(218,231,245);text-align: center;color:
rgb(22,17,32)"> <th> </th> <th>A</th> <th>B</th> <th>C</th> <th>D</th> <th>E</th> <th>F</t
h> </tr> </thead>
...
```

✦ 18.6 Split a string into a sequence of characters



SPL script is as follows, where A.split() function is used to split the strings into a sequence of characters :

	A	B	C	B
1	=file("code.html").read()			/Read the file as a string
2	=A1.split()	0	0	/A.split() splits the string into a sequence of characters
3	for A2	if A3=="["	=B2+=1	/If the left bracket appears, B2 plus 1. It is used to match the bracket
4		else if A3=="]"	=B2-=1	/If the right bracket appears, B2 minus 1. It is used to match the bracket
5		else if A3==","&&B2==0	>C2+=1	/If a comma appears and the brackets match, add 1 to C2'count. C2 is the number of commas outside brackets

A2	Members
	<
	h
	t
	m
	...

C2	Value
	27

✦ 18.7 Split strings into a sequence of words



Split strings into a sequence of words.

Count the top three words in terms of frequencies in an article. Part of the article is as follows:

How to Call an SPL Script in Java

esProc provides its own JDBC driver to become integration-friendly with a Java application. The method of calling an esProc SPL script is similar to the execution of SQL queries and stored procedures in Java.

Deploying esProc JDBC in a Java application

Simply put, to deploy JDBC in a Java application is to put in place the necessary jars and configuration files for loading esProc when starting the application. esProc JDBC requires JDK 1.6 or a higher version.

...

✦ 18.7 Split strings into a sequence of words



A.words() function splits the string into a sequence of words. SPL script is as follows:

	A	B
1	=file("callSPL.txt").read()	/Read the file as a string
2	=A1.words()	/ A.words() function splits the words away from the string
3	=A2.group()	/Group words
4	=A3.ptop(-3;~.len())	/Select the top three words that appear most frequently
5	=A3(A4).(~(1)).concat@c()	/Concatenate the top three words into a string with commas

A5	Value
	the,property,name

✦ 18.8 Use tab as a separator to split a string



Use tab as a separator to split a string into a sequence of strings.

Organize a log file into structured data (a table sequence consisting of fields of USERID,UNAME,IP,TIME,URL,BROWSER,LOCATION, and MODULE) . Log format: the first line is IP, TIME, GET, URL, BROWSER; the second line is MODULE; the third line is USERID, UNAME, LOCATION.

10.10.10.143	2013-04-01 21:14:44	GET	/p/pt301/index.jsp	Mozilla/6.0
#module:production#				
47356	Jessica	Chicago		
10.10.2.76	2013-04-01 21:18:50	GET	/h/homepage.jsp	Chrome/35
#module:homepage#				
419	Jacob	Houston		
10.10.54.218	2013-04-01 21:25:19	GET	/p/pt27/index.jsp	Mozilla/6.0
#module:production#				
3464	Madison	Detroit		
10.10.10.145	2013-04-01 21:30:02	GET	/u/userlist/showlist.jsp	Mozilla/6.0
#module:usercenter#				
432442	Phoenix	San Jose		
10.2.1.242	2013-04-01 21:30:15	GET	/p/pt271/index.jsp	Mozilla/6.0
#module:production#				
3435567	Megan	San Jose		

✦ 18.8 Use tab as a separator to split a string



s.split(d) function splits the string s into sequences through the separator d. SPL script is as follows:

	A	B
1	=file("log.txt").read@n()	/Read the file as a sequence of strings by line
2	=A1.group((#-1)\3)	/Use group function to group every three rows
3	=A2.(~.conj(~.split("\t")))	/Use s.split() function to split each line in each group by "\t" and merges results into a sequence
4	=A3.new(~(7):USERID,~(8):UNAME,~(1):IP,~(2):TIME,~(4):URL,~(5):BROWSER,~(9):LOCATION,left(~(6).split(":")(2),-1):MODULE)	/Generate structured data

A4							
USERID	UNAME	IP	TIME	URL	BROWSER	LOCATION	MODULE
47356	Jessica	10.10.10.143	2013-04-01 21:14:44	/p/pt301/index.jsp	Mozilla/6.0	Chicago	production
419	Jacob	10.10.2.76	2013-04-01 21:18:50	/h/homepage.jsp	Chrome/35	Houston	homepage
...

✦ 18.9 Use comma as the separator to split a string



Use comma as the separator to split a string into a sequence of strings.

Query names of the products purchased by customers, and separate multiple names with commas. Part of the product table and sales table is as follows:

Product		
ID	Name	Website
R	Report	http://www.raqsoft.com.cn/r
P	esProc	http://www.raqsoft.com.cn/p
C	esCalc	http://www.raqsoft.com.cn/c
M	AI Models	http://www.yimming.com/
...

Sales		
ID	Customer	Product
1	VINET	R
2	TOMSP	P,R
3	HANAR	P,R,C
4	VICTE	P
...

✦ 18.9 Use comma as the separator to split a string



s.split(d) function uses @c option to do the splitting. When d is omitted, split the string into single characters. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Product")	/Read product table
3	=A1.query("select * from Sales")	/Read sales table
4	=A3.run(Product=Product.split@c())	/Use @c option with the split function to split the products in sales table into a sequence by comma to return
5	=A4.run(Product=Product.(A2.find(~).Name).concat@c())	/Get a sequence of product names by product ID, and concatenate the members into a string with commas

A4		
ID	Customer	Product
1	VINET	[R]
2	TOMSP	[P,R]
3	HANAR	[P,R,C]
...

A5		
ID	Customer	Product
1	VINET	Report
2	TOMSP	esProc,Report
3	HANAR	esProc,Report,esCalc
...

✦ 18.10 Split a string into two segments by specified separator



Use "?" or "=" as the separator to split a string into two strings.

A website records the URLs visited by users. Query the most frequently used search criteria. Some contents are as follows:

ID	User	Website
1	Rebecca	https://github.com/search?q=How+to+study+java%3F
2	Ashley	https://github.com/search?q=report&type=Code
3	Rachel	https://github.com/search?q=bigdata&type=Repositories
4	Rachel	https://github.com/search?l=Python&q=bigdata&type=Repositories
...

✦ 18.10 Split a string into two segments by specified separator



Use @1 option with s.split(d) function to find the first d and stop searching, which splits a string into two segments. SPL script is as follows:

	A	B
1	=file("loginUrls.txt").import@t()	/Read user login file
2	=A1.(Website.split@1("?")(2))	/Use @1 option with s.split() function to split a string into two segments according to the first ?
3	=A2.(~.split("&").select@1(like(~,"q=*")))	/Split the parameter by & and select the criteria of q = * , which is the user search criteria
4	=A3.(~.split@1("=")(2))	/Split the search criteria into two segments by = and the second part is the search criteria
5	=A4.group()	/Group by search criteria
6	=A5.maxp(~.len()(1))	/Select the group with the largest number of members, which is the most frequently used search criteria

A6

Value

bigdata

✦ 18.11 Split a string with regular expression



Use a regular expressions to split a string into a sequence of strings.

Remove all comments (<!-- -->) from the HTML file. Part of the contents are as follows:

```
<html>
<!-- Row Highlight Javascript -->
<script type="text/javascript">
    window.onload=function(){
    var tfrow = document.getElementById('tfhover').rows.length;
    var tbRow=[];
    ...
};
...
</html>
```


✦ 18.11 Split a string with regular expression



@r option is used with s.split(d) function. d is interpreted as a regular expression. SPL script is as follows:

	A	B
1	=file("table.html").read()	/Read the html file
2	=A1.split@r("<!--.*-->")	/Use @r option with s.split() function to split the string according to the regular expression
3	=A2.concat()	/Concatenate the strings after splitting, that is, the HTML format string without comments
4	>file("table.html").write(A3)	/Write the string to a file

A3

Value

```
<html><script type="text/javascript">window.onload=function(){...
```

✦ 18.12 Parse a string into numerical value



Parse a string into numerical value.

The model performance table records various indexes of different models. We want to select the numerical target model (ModelType is 2) and present it with the indexes as the column names. Part of the data is as follows:

ID	ModelName	ModelType	Performance
1	HousePrice	2	SquareR=0.933743
2	HousePrice	2	MSE=295749426.986263
3	HousePrice	2	RMSE=17197.366862
4	HousePrice	2	GINI=0.197449
5	HousePrice	2	MAE=12509.456071
6	HousePrice	2	MAPE=7.798386
7	Titanic	1	GINI=0.654867
8	Titanic	1	AUC=0.827434
9	Titanic	1	KS=0.587658
...

✦ 18.12 Parse a string into numerical value



number(stringExp) function is used to parse the string stringExp into a numeric value. SPL script is as follows:

	A	B
1	=file("mps.txt").import@t()	/Import model performance file
2	=A1.select(ModelType:2)	/Select model type 2
3	=A2.group(ModelName)	/Group by model name
4	=A3(1).(Performance.split("=")(1)).concat@c()	/Concatenate the first group of index names into a string with commas
5	=create(\$"modelName,"+A4})	/Create an empty table sequence. The first column is the model name, followed by the model performance indexes
6	=A3.(A5.record(A3.~.modelName A3.~.(number(Performance.split("=")(2))))))	/Insert indexes into A5's table sequence in loop. Here we use the number() function to convert the split string into a numeric value

A5

modelName	SquareR	MSE	RMSE	GINI	MAE	MAPE
HousePrice	0.933743	295749426.986263	17197.366862	0.197449	12509.456071	7.798386
...

✦ 18.13 Parse a percentage string into a numerical value



Parse a percentage string into a numerical value.

Based on the prediction results of Titanic survival probability model, calculate the proportion of female among people with a survival probability over 80%. Part of the data is as follows:

Survived	PassengerId	Pclass	Name	Gender	...
Percent:10.461%	624	3	Braund, Mr. Owen Harris	male	...
Percent:9.108%	625	3	Cumings, Mrs. John Bradley	male	...
Percent:8.891%	626	1	Heikkinen, Miss. Laina	male	...
Percent:50.510%	627	2	Futrelle, Mrs. Jacques Heath	male	...
...

✦ 18.13 Parse a percentage string into a numerical value



`number(stringExp, format)` function is used to parse the string *stringExp* into a numerical value according to the specified *format*. SPL script is as follows:

	A	B
1	<code>=file("titanic.csv").import@cqt()</code>	/Import Titanic data file
2	<code>=A1.run(Survived=Survived.split(":")(2))</code>	/Split the survival probability field by ":" and get the second part
3	<code>=A2.run(Survived=number(Survived, "0%"))</code>	/Use number() function to parse the value according to the specified format
4	<code>=A3.select(Survived > 0.8)</code>	/Select people with a survival probability of more than 80%
5	<code>=string(A4.count(Gender=="female") / A4.len() , "0.000%")</code>	/Calculate the proportion of females

A5

Value

97.619%

✦ 18.14 Automatically parse a string into the proper data type



Parse a string into the proper data type automatically.

According to the Olympic medal table, find the Olympic Games where China ranks higher than Russia.

Game	Nation	Medal
30	USA	[46,29,29]
30	China	[38,27,23]
30	UK	[29,17,19]
30	Russia	[24,26,32]
30	Korea	[13,8,7]
...

✦ 18.14 Automatically parse a string into the proper data type



parse(s) function parses string s into the proper data type. SPL script is as follows:

	A	B
1	=file("Olympic.csv").import@cqt()	/Import Olympic games medal table
2	=A1.run(Medal=parse(Medal))	/Use parse() function to parse the medal field into a sequence
3	=A2.group(Game)	/Group by Game
4	=A3.select(~.select(Nation=="China").Medal>~.select(Nation=="Russia").Medal)	/Use ">" to compare the sequences of medals for China and Russia by comparing the number of gold, silver and bronze in order, and select the games where China ranks higher.
5	=A4.(Game)	/List the games

A5	Game
	23
	25
	28
	29
	30

✦ 18.15 Split a string and parse the split members into proper data types



While splitting a string, parse the split members into the proper data types.

There are course table and course selection table. Find the courses not selected by students. Multiple courses can be selected that are separated by commas. Part of the data is as follows:

Course		
ID	NAME	TEACHERID
1	Environmental protection and ...	5
2	Mental health of College Students	1
3	Computer language Matlab	8
4	Electromechanical basic practice	7
5	Introduction to modern life science	3
6	Modern wireless communication system	14
...

SelectCourse		
ID	STUDENTID	COURSE
1	59	2,7
2	43	1,8
3	52	2,7,10
4	44	1,10
5	37	5,6
6	57	3
...

✦ 18.15 Split a string and parse the split members into proper data types



@p option is used with s.split() function to parse the split members into proper data types. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Course")	/Read course table
3	=A1.query("select * from SelectCourse")	/Read course selection table
4	=A3.union(COURSE.split@cp())	/Use @p and @c options with split function to split the courses in the course selection table by commas, parse them into integers, and get the union of the sequences of courses by union() function
5	=A2.(ID)	/Get IDs of all courses
6	=A2(A5.pos([A5,A4].diff()))	/Use diff() function to find the difference of course IDs in course table and course selection table, that is, the unselected course. After locating the ID in A5, select the corresponding record from A2

A6		
ID	NAME	TEACHERID
1	Fundamentals of economic management	21

✦ 18.16 Parse string to table sequence



Parse a string into a table sequence.

The GDP and population data of major cities in China are copied from the system's clipboard. Export them to a file of CSV format. Part of the contents are as follows:

ID	City	GDP	Population
1	Shanghai	32679	2418
2	Beijing	30320	2171
3	Shenzhen	24691	1253
4	Guangzhou	23000	1450
5	Chongqing	20363	3372
6	Tianjin	18809	1557
7	Suzhou	18597	1068
8	Chengdu	15342	1605
...			

✦ 18.16 Parse string to table sequence



`S.import (;s)` function is used to import the contents read from string `S` as records and return them as a table sequence, where `s` is the separator (default is tab). `@t` option means that reading the first line as the title. SPL script is as follows:

	A	B
1	<code>=clipboard()</code>	/Return the contents of the clipboard as strings
2	<code>=A1.import@t()</code>	/Read the strings into a table sequence. The separator is tab (<code>\t</code>) by default. <code>@t</code> option means that the first line is the title
3	<code>>file("GDP.csv").export@ct(A2)</code>	/Export A2's table sequence to <i>GDP.csv</i> file

GDP.csv

```
ID, City, GDP, Population
1, Shanghai, 32679, 2418
2, Beijing, 30320, 2171
3, Shenzhen, 24691, 1253
4, Guangzhou, 23000, 1450
5, Chongqing, 20363, 3372
...
```

✦ 18.17 Parse the string type field in a table sequence with regular expression



Parse the character string type field in a table sequence with regular expression.

Get the number from customer address. Part of the customer table is as follows:

ID	Name	City	Address
1	VINET	Beijing	124 Guangming North Road
2	TOMSP	Jinan	543 Qingnian East Road
3	HANAR	Qinhuangdao	22 Guanghai Street
4	VICTE	Nanjing	Qinglin bridge 68
...

✦ 18.17 Parse the string type field in a table sequence with regular expression



S.regex(rs) function searches for matching section in string *S* with regular expression *rs*, and returns null if no matching part is found. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Customer")	/Read customer table
3	=A2.run(Address=number(Address.regex("\\D*(\\d+)\\D*")(1)))	/Use S.regex() function to get the street number from the address and parse it into number

A3	ID	Name	City	Address
	1	VINET	Beijing	124
	2	TOMSP	Jinan	543
	3	HANAR	Qinhuangdao	22

✦ 18.18 Parse indefinite-structure text with regular expression



Parse a text file with indefinite number of lines with the regular expression.

The log file contains an indefinite number of lines. Now we want to parse it into structured data. Part of the log file is as follows:

```
1 Window Report for "aaa"
2
3 Object Type: Symbol
4     Location: left: 695 top: 51 right: 723 bottom: 75
5
6     Line Color:          RGB ( 0 0 0 )
7     Fill Color:         RGB ( 255 255 0 )
8
9     Link:
10    Type: Push Button - Action Script
11    Condition Type: On Left Click/Key Down
12
13    Statement:
14        CALL AufrufSchalter( VEA_450P02.Name, SchalterH-0-A);
15
16
17    Link:
18    Type: Disable - Discrete Expression
19    Expression :AccessLevelBedienenVEZ ==1
20
21
22    Disabled When Expression is FALSE
23
24 Object Type: Polygon
25     Location: left: 695 top: 56 right: 700 bottom: 60
26
27     Line Color:          RGB ( 0 0 0 )
28     Fill Color:         RGB ( 229 229 229 )
```

✦ 18.18 Parse indefinite-structure text with regular expression



A.regex(rs,Fi) function searches for matching sections in the string members of sequence A with the regular expression rs, and returns the results to form a table sequence with Fi as the fields. SPL script is as follows:

	A	B
1	=file("report.log").read()	/Read the log file and return as string
2	=A1.split("Object Type:").delete(1)	/Split the text content into multiple records according to the mark "Object Type:" and discard the first record
3	=A2.regex("(.)[\\s\\S]+left:(.)[\\s\\S]+top:(.)[\\s\\S]+right:(.)[\\s\\S]+bottom:(.)[\\s\\S]+Line Color:(.)[\\s\\S]+Fill Color:\\t\\t(.+)[\\S\\s]+Link:(.)[\\s\\S]+Type: (.+)[\\s\\S]+Condition Type:(.)[\\s\\S]+Statement:\\s+(.+)[\\s\\S]+Link:(.)[\\s\\S]+Type: (.+)[\\s\\S]+Expression :(.+)";ObjectType,left,top,right,bottom,lineColor,fillColor,objctLink,type,conditionType,statement,statementLink,statementType,lastExpress)	/For each member, search for matching part with regular expression, and concatenate the results into a record
4	=file("result.txt").export@t(A3)	/Export A3's result to <i>result.txt</i>

A3	ID	ObjectType	left	top	right	bottom	lineColor	fillColor	...
	1	Symbol	695	51	723	75	RGB (0 0 0)	RGB (255 255 0)	...

✦ 18.19 Use code to parse string type fields in a table sequence



Use code to parse character string type fields in a table sequence.

Find the average salary of employees who were born in the 1980s. The age needs to be extracted from the ID number. Part of the employee table is as follows:

ID	Name	Identification	Salary
1	Rebecca	Driving license:495319197411204628	7000
2	Ashley	ID number:103263198007194980	11000
3	Rachel	ID number:721125197012173641	9000
4	Emily	ID number:619124198503071617	7000
5	Ashley	ID number:248238197505138795	16000
...

✦ 18.19 Use code to parse string type fields in a table sequence



When a single function cannot solve the problem directly, multiple functions can be used to parse and process strings step by step. SPL script is as follows:

	A	B
1	=connect("db").query("select * from Employee")	/Connect to database and read employee table
2	=A1.run(Identification=Identification.regex("\\D*(\\d+)")(1))	/Use S.regex() function to read the number part of ID
3	=A2.run(Identification=mid(Identification,7,4))	/Use mid() function to read No.7 to No.10 digits of the ID, that is, the year of birth
4	=A3.run(Identification=number(Identification))	/Use number() function to parse the year string into a number
5	=A4.select(Identification>=1980 && Identification <=1989)	/Select employees born in the 1980s
6	=A5.avg(Salary)	/Calculate the average salary

A6	ID
	7256.16

✦ 18.20 Modify the filter condition in the SQL statement



Modify the filter condition in the SQL statement.

The following SQL statement is used to select the employees in sales department whose salary is greater than 10000. Modify the department in the filter condition to the R&D department.

```
select
    EID,NAME,SURNAME,DEPT,SALARY
from
    Employee
where
    DEPT='sales' and SALARY>10000
```

✦ 18.20 Modify the filter condition in the SQL statement



s.sqlparse(part) function is used to split SQL into a sequence of individual parts. The part parameter is used to replace the corresponding part of SQL and return the new SQL. @w option represents the where statement and @s represents the select statement. SPL script is as follows:

	A	B
1	<code>select EID,NAME,SURNAME,DEPT,SALARY from Employee where DEPT='sales' and SALARY>10000</code>	/Define SQL constants
2	<code>=A1.sqlparse@w()</code>	/Use @w option with s.sqlparse() function to get the where condition
3	<code>=A2.split@t("and")</code>	/Use s.split() function to split the where condition; @t option performs trim over each segment
4	<code>=A3.pselect(like(~,"DEPT*"))</code>	/Select the department condition
5	<code>=A3(A4)="DEPT='R&D'"</code>	/Change the department condition to R&D
6	<code>=A3.concat(" and ")</code>	/Concatenate members in the sequence of conditions with and
7	<code>=A1.sqlparse@w(A6)</code>	/Use @w option with s.sqlparse(part) function to replace the where condition

A6

Value

`select EID,NAME,SURNAME,DEPT,SALARY from Employee where DEPT='R&D' and SALARY>10000`

✦ 18.21 Translate standard SQL statements into specified database format



Translate standard SQL statements into the format used by the specified database.

The sales data of a company is stored in two databases, Oracle and MySQL. Find the number of orders with sales over 1000 during the period from March 18 to July 18, 2015. The two database tables have same structure, as shown below:

ORDERID	CUSTOMERID	EMPLOYEEID	ORDERDATE	AMOUNT
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 18.21 Translate standard SQL statements into specified database format



sql.sqltranslate(dbtype) function translates functions in standard SQL into the format of the specified database. SPL script is as follows:

	A	B
1	select ORDERID,CUSTOMERID,EMPLOYEEID,ORDERDATE,AMOUNT from ORDERS where ORDERDATE between date('2015-03-18') and date('2015-07-18') and AMOUNT>1000	/Standard SQL
2	=A1.sqltranslate("ORACLE")	/Translate standard SQL into Oracle format
3	=A1.sqltranslate("MYSQL")	/Translate standard SQL into MYSQL format
4	=connect("oracle").query(A2)	/Connect Oracle and execute SQL
5	=connect("mysql").query(A3)	/Connect MySQL and execute SQL
6	=A4,A5].merge@ou(ORDERID)	/Order-based merge, during which orders with same ID are removed
7	=A6.len()	/Count the number
A7	Value	
	63	

✦ 18.21 Translate standard SQL statements into specified database format



Standard SQL (A1) :

select ORDERID,CUSTOMERID,EMPLOYEEID,ORDERDATE,AMOUNT from ORDERS where
ORDERDATE between **date**('2015-03-18') and **date**('2015-07-18') and AMOUNT>1000

ORACLE (A2) :

select ORDERID,CUSTOMERID,EMPLOYEEID,ORDERDATE,AMOUNT from ORDERS where
ORDERDATE between **TO_DATE**('2015-03-18','YYYY-MM-DD') and **TO_DATE**('2015-07-18','YYYY-MM-DD') and AMOUNT>1000

MYSQL (A3) :

select ORDERID,CUSTOMERID,EMPLOYEEID,ORDERDATE,AMOUNT from ORDERS where
ORDERDATE between **DATE_FORMAT**('2015-03-18','%Y-%m-%d') and **DATE_FORMAT**('2015-07-18','%Y-%m-%d') and AMOUNT>1000

✦ 18.22 Parse and analyze HTML file



Parse HTML file and analyze the text.

Find the numbers in the body of an HTML file. Part of the contents of the file are as follows:

```
<!DOCTYPE html>
<html class="html__responsive html__unpinned-leftnav">
<head>
  <title>Stack Overflow - Where Developers Learn, Share, & Build Careers</title>
    <link rel="shortcut icon"
href="https://cdn.sstatic.net/Sites/stackoverflow/Img/favicon.ico?v=ec617d715196">
    <link rel="apple-touch-icon" href="https://cdn.sstatic.net/Sites/stackoverflow/Img/apple-
touch-icon.png?v=c78bd457575a">
    <link rel="image_src" href="https://cdn.sstatic.net/Sites/stackoverflow/Img/apple-touch-
icon.png?v=c78bd457575a">
  ...
</html>
```

✦ 18.22 Parse and analyze HTML file



s.htmlparse() function gets all the text in an HTML file. SPL script is as follows:

	A	B
1	=file("sof.html").read()	/Read the html file
2	=A1.htmlparse()	/Use htmlparse() function to parse the HTML strings and return a sequence of all the text
3	=A2.(~.words@d()).conj()	/Calculate the parsed sequence of text in loop, get the number in each string, and then calculate the their concatenation

A3	Members
	30
	3
	16.5
	5
	...

✦ 18.23 Parse HTML file to get table sequence



Parse an HTML file to generate a table sequence.

Parse the HTML file below to get the score table, and count the total score of each student.

```
<html>
...
<table id="tfhover" class="tftable" border="1">
<tr> <th>CLASS</th> <th>STUDENTID</th> <th>SUBJECT</th> <th>SCORE</th> </tr>
<tr> <td>Class one</td> <td>1</td> <td>Math</td> <td>77</td>
<tr> <td>Class one</td> <td>1</td> <td>PE</td> <td>69</td>
<tr> <td>Class one</td> <td>1</td> <td>English</td> <td>84</td>
<tr> <td>Class one</td> <td>2</td> <td>Math</td> <td>80</td>
<tr> <td>Class one</td> <td>2</td> <td>PE</td> <td>97</td>
...
</table>
...
</html>
```

✦ 18.23 Parse HTML file to get table sequence



s.htmlparse(tag:i;j) function gets the j^{th} text under the i^{th} tag in HTML format string s. SPL script is as follows:

	A	B
1	=file("table.html").read()	/Read the html file
2	=A1.htmlparse("table":0)	/Use htmlparse() function to parse the HTML string and return all the contents of the first table tag
3	=create(\${A2(1).concat@c()})	/Create a table sequence table with the heading in the first row
4	=A3.record(A2.to(2,).conj())	/Insert the data starting from the second row in turn into A3's table sequence
5	=A3.groups(STUDENTID; sum(SCORE):TOTALSCORE)	/Group and aggregate the student score table, and calculate the total score of each student

A3	STUDENTID	TOTALSCORE
	1	230
	2	258
	3	228

✦ 18.24 Calculate the date N days after a certain date



Calculate the date N days after a certain date.

Find the orders of 2015 delivered on the second day and arrived within three days after the delivery. The orders table is as follows:

ID	CustomerID	OrderDate	DeliveryDate	ArrivalDate	Amount
10248	VINET	2012/07/04	2012/07/16	2012/08/01	428.0
10249	TOMSP	2012/07/05	2012/07/10	2012/08/16	1842.0
10250	HANAR	2012/07/08	2012/07/12	2012/08/05	1523.5
10251	VICTE	2012/07/08	2012/07/15	2012/08/05	624.95
10252	SUPRD	2012/07/09	2012/07/11	2012/08/06	3559.5
...

✦ 18.24 Calculate the date N days after a certain date



Use `date + n` to calculate the date on the n^{th} day after the specified date. SPL script is as follows:

	A	B
1	<code>=connect("db")</code>	/Connect to database
2	<code>=A1.query("select * from Orders")</code>	/Read the orders table
3	<code>=A2.select(year(OrderDate)==2015 && OrderDate+1>=DeliveryDate && DeliveryDate+3>=ArrivalDate)</code>	/Use the symbol "+" to calculate the date on the n^{th} day after the specified date.

A3	ID	CustomerID	OrderDate	DeliveryDate	ArrivalDate	Amount
	11094	BERGS	2015/07/18	2015/07/18	2015/07/19	506.05
	11101	AROUT	2015/07/18	2015/07/18	2015/07/20	130.0
	11102	AROUT	2015/07/18	2015/07/19	2015/07/20	240.0

✦ 18.25 Calculate the number of days between two dates



Calculate the number of days between two dates.

Find orders in 2015 whose delivery date is over 30 days after they are created. The orders table is as follows:

ID	CustomerID	OrderDate	DeliveryDate	Amount
10248	VINET	2012/07/04	2012/07/16	428.0
10249	TOMSP	2012/07/05	2012/07/10	1842.0
10250	HANAR	2012/07/08	2012/07/12	1523.5
10251	VICTE	2012/07/08	2012/07/15	624.95
10252	SUPRD	2012/07/09	2012/07/11	3559.5
...

✦ 18.25 Calculate the number of days between two dates



The symbol "-" is used to calculate the number of days between the two dates. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Orders")	/Read the orders table
3	=A2.select(year(OrderDate)=2014 && DeliveryDate-OrderDate>30)	/Use the symbol "-" to calculate the number of days between the delivery date and the order date

A3	ID	CustomerID	OrderDate	DeliveryDate	Amount
	10924	BERGS	2014/03/04	2014/04/08	1835.7
	10927	LACOR	2014/03/05	2014/04/08	800.0
	10970	BOLID	2014/03/24	2014/04/24	224

✦ 18.26 Calculate the number of seconds / minutes between two datetimes



Calculate the number of seconds/minutes between two datetimes.

For each ID, accumulate values from the first time that 1 appears until 0 appears. If 0 does not appear, compare the value with the system time. Part of the data is as follows:

ID	Time	Value
1	2020/07/08 15:00:00	1
1	2020/07/08 15:02:00	1
1	2020/07/08 15:04:00	1
1	2020/07/08 15:06:00	0
1	2020/07/08 15:08:00	0
1	2020/07/08 15:10:00	1
1	2020/07/08 15:20:00	0
2	2020/07/08 15:02:00	1

✦ 18.26 Calculate the number of seconds / minutes between two datetimes



now() function gets the current system date time. interval (datetimeExp1,datetimeExp2) function calculates the interval between two date time type data, where @s option returns the interval in the unit of seconds. SPL script is as follows:

	A	B
1	=file("table.txt").import@t()	/Read the file
2	=A1.group(ID).(~.group@o1(Value))[null]	/Group by ID, then in each group, perform the merge grouping by checking whether the adjacent values are same, and get the first record of each subgroup. Add a null value to each group for the convenience of subsequent calculation
3	=A2.news(~.len()\2;ID,(s=A2.~(#*2-1).Time):StartTime, interval@s(s,ifn(A2.~(#*2).Time,now()))/60:CumTime)	/In each group, make the time in the odd row the start time and the time of corresponding even row the end time to calculate the interval duration. If the corresponding even row is null, make the current system time the end time to calculate the interval duration

A3	ID	StartTime	CumTime
	1	2020/07/08 15:00:00	6.0
	1	2020/07/08 15:10:00	10.0
	2	2020/07/08 15:02:00	28.0

✦ 18.27 Calculate the first day and last day of the week



Calculate the first day and last day of the week.

The current date is 2020/02/17. Calculate the growth rate of the SSE Composite Index last week. Part of the data is as follows:

Date	Open	Close	Amount
2020/02/17	2924.9913	2983.6224	3.67E11
2020/02/14	2899.8659	2917.0077	3.08E11
2020/02/13	2927.1443	2906.0735	3.35E11
2020/02/12	2895.5561	2926.8991	2.98E11
2020/02/11	2894.5414	2901.6744	3.03E11
...

✦ 18.27 Calculate the first day and last day of the week



pdate(dateExp) function gets the first day and the last day of the week / month / quarter that the specified date dateExp belongs to. SPL script is as follows:

	A	B
1	=file("sh000001.csv").import@cqt()	/Read SSE Composite Index data
2	=A1.sort(Date)	/Sort by date
3	=pdate@w(A2.m(-1).Date)	/Use @w option with pdate() function to select the first day (Sunday) of the week that the day (2020/ 02/17) belongs to
4	=A2.select@z1(Date<=A3-2)	/Find the first record before last Friday from back to front, that is, the last record of the previous trading week
5	=pdate@w(A4.Date)	/Find the first day of the previous trading week (Sunday)
6	=A2.select@z1(Date<=A5-2)	/Find the first record of the previous Friday before last trading week from back to front, that is, the last record of the week before the previous trading week
7	=A4.Close/A6.Close-1	/Calculate the growth rate

A7	Value
	0.01427

✦ 18.28 Calculate the average daily sales for a quarter



Calculate the average daily sales for each quarter.

Calculate the average daily sales for each quarter in 2014. Part of the data in the sales table is as follows:

ORDERID	CUSTOMERID	EMPLOYEEID	ORDERDATE	AMOUNT
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 18.28 Calculate the average daily sales for a quarter



days(dateExp) function gets the number of days in the year, quarter or month that the specified date dateExp belongs to. @q option is used to get the number of days in the quarter that the specified date belongs to. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Sales")	/Read the sales table
3	=A2.select(year(OrderDate)=2014)	/Select records of 2014
4	=A3.groups((month(OrderDate)+2)\3:Quarter; sum(Amount):Amount)	/Group and aggregate by quarter, and calculate the total sales of each quarter
5	=A4.run(Amount=Amount / days@q(date("2014/"/(Quarter*3)+"/01"))))	/Use days() function to calculate the number of days of each quarter, and divide the total sales by the number of days to calculate the average daily sales

A5	Quarter	Amount
	1	1765.33
	2	1764.96
	3	2034.56
	4	2355.63

✦ 18.29 Calculate age



Calculate age based on birth date.

Find the average age of employees in each department. The employee table is as follows:

ID	NAME	BIRTHDAY	HIREDATE	DEPT	SALARY
1	Rebecca	1974/11/20	2005/03/11	R&D	7000
2	Ashley	1980/07/19	2008/03/16	Finance	11000
3	Rachel	1970/12/17	2010/12/01	Sales	9000
4	Emily	1985/03/07	2006/08/15	HR	7000
5	Ashley	1975/05/13	2004/07/30	R&D	16000
...

✦ 18.29 Calculate age



age(x) function calculates the number of years from x to the current date. SPL script is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Employee")	/Read employee table
3	=A1.groups(DEPT; avg(age(BIRTHDAY)):AvgAge)	/Group records by department and calculate of the average age of each department. The age() function is used to calculate the age of employees

A3	DEPT	AvgAge
	Administration	43.5
	Finance	38.83
	HR	41.05

✦ 18.30 Calculate the date N months before a certain date



Calculate the date N months before a certain date.

Query the total sales amount in the three months before May 21, 2014. Part of the data in the sales table is as follows:

ORDERID	CUSTOMERID	EMPLOYEEID	ORDERDATE	AMOUNT
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 18.30 Calculate the date N months before a certain date



`elapse(dateExp, n)` function calculates the new date with a certain time difference from the specified date. If `n` is a negative number, it gets a new date before `n` days / years / months. `@m` option is used to calculate a new date that differs by `n` months from the specified date. SPL script is as follows:

	A	B
1	<code>=connect("db")</code>	/Connect to database
2	<code>=A1.query("select * from Sales")</code>	/Read sales table
3	<code>=date("2014/05/21")</code>	/Define a date
4	<code>=A2.select(OrderDate>=elapse@m(A3,-3) && OrderDate<A3)</code>	/Use <code>elapse()</code> function to calculate the date 3 months before A3's date. Select data for the first three months from the sales table
5	<code>=A4.sum(Amount)</code>	/Calculate the total sales amount

A5	Value
	154074.49

✦ 18.31 Calculate the date after N working days



Calculate the date after N working days.

Calculate how many times that each employee has had when they cannot solve customer's problems for more than 10 working days in 2014. Part of the data is as follows:

ID	CustomerID	EmployeeId	QuestionDate	SolveDate
1	OLDWO	2	2014/01/01	2014/01/09
2	WELLI	7	2014/01/01	2014/01/07
3	LAUGB	2	2014/01/01	2014/01/07
4	LINOD	8	2014/01/02	2014/01/08
5	REGGC	5	2014/01/02	2014/01/12
...

✦ 18.31 Calculate the date after N working days



workday (t, k, h) function calculates the date k working days away from the date t. h is a sequence of (non) holidays, which means that if a date member is not a weekend, it's treated as a holiday. If a member is weekend, it is treated as the working day. SPL script is as follows:

	A	B
1	=file("AfterSale.csv").import@ct()	/Import AfterSale table
2	[2014/01/01,2014/01/26,2014/01/31,2014/02/03,2014/02/04,2014/02/05,2014/02/06,2014/02/08,2014/04/07,2014/05/01,2014/05/02,2014/05/04,2014/06/02,2014/09/08,2014/09/28,2014/10/01,2014/10/02,2014/10/03,2014/10/06,2014/10/07,2014/10/11]	/Define holidays in 2014
3	=A1.select(year(QuestionDate)=2014 && workday(QuestionDate, 10, A2) < SolveDate)	/Use workday() function to calculate the date after 10 working days by skipping the holidays
4	=A3.groups(EmployeeId; count(~):Count)	/Group and aggregate by employee and count times

A4	EmployeeID	Count
	1	2
	2	1
	3	2

✦ 18.32 Get a sequence of working days



List the sequence of working days between two dates.

List the names of personnel on duty in each working day from 2020/04/27 to 2020/05/08. Part of the contents of attendance table are as follows:

ID	Date	Name
1	2020/04/27	Emily
2	2020/04/28	Emily
3	2020/04/28	Johnson
4	2020/04/29	Emily
5	2020/04/30	Johnson
...

✦ 18.32 Get a sequence of working days



workdays(b, e, h) function generates a sequence of working days between date b and date e, including b and e. h is a sequence of (non) holidays, which means that if a date member is not a weekend, it's treated as a holiday. If a member is weekend, it is treated as the working day. SPL script is as follows:

	A	B
1	[2020/04/27,2020/05/08]	/Define start date and end date
2	=workdays(A1(1),A1(2),[date("2020/05/01")])	/workdays() function gets the working days in the interval, excluding May 1, which is a holiday
3	=file("Duty.txt").import@t()	/Import duty table
4	=A3.align@a(A2, Date)	/Group duty table in alignment with the sequence of working days, and all records are matching in each group
5	=A4.new(~.Date:Date, ~.(Name).concat@c():Names)	/Create a table sequence and concatenate the names in each group with commas

A5	Date	Names
	2020/04/27	Emily
	2020/04/28	Emily,Johnson
	2020/04/29	Emily

✦ 18.33 Get a sequence of dates between two dates



List the date sequence between two dates.

When overlapping parts are not counted repeatedly, calculate the total number of days contained in multiple time periods. Part of the data is as follows:

ID	Start	End
1	2012/07/04	2012/07/16
2	2012/07/06	2012/07/10
3	2012/07/19	2012/07/24
4	2012/07/22	2012/07/25
5	2012/07/30	2012/08/02
...

✦ 18.33 Get a sequence of dates between two dates



periods(s, e, i) function returns a sequence of time values spaced a specified period (i) apart from each other from s to e (including endpoints). The default unit is day; i is 1 by default. SPL script is as follows:

	A	B
1	=file("periods.txt").import@t()	/Read periods from file
2	=A1.(periods(Start,End))	/Calculate the dates contained in each period in loop
3	=A2.union()	/Find the union of dates
4	=A3.len()	/Calculation the number of days

A2
Members
[2012/07/04,2017/07/05,...]
[2012/07/06,2017/07/07,...]
[2012/07/19,2017/07/20,...]
...

Members
2017/07/06
2017/07/07
2017/07/08
2017/07/09
2017/07/10

A4
Value
52

✦ 18.34 Divide the period between two dates equally into n segments



Divide the period between two dates equally into n segments.

Divide sales records from January 20, 2014 to January 20, 2015 (not included) into 4 groups according to dates and store the groups in files respectively. Part of the the sales table is as follows:

ORDERID	CUSTOMERID	EMPLOYEEID	ORDERDATE	AMOUNT
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
10404	MAGAA	2	2014/01/03	1675.0
...

✦ 18.34 Divide the period between two dates equally into n segments



range (s,e,k:n) function divides the interval between s and e equally into n parts, returns the header of k and k+1 segment as a two-members-sequence. SPL script is as follows:

	A	B	C
1	=file("Sales.txt").import@qt()		/Import sales table
2	[2014/01/20,2015/01/20]		/Define start date and end date
3	for 4	=range(A2(1),A2(2), A3:4)	/Perform loop operation that uses the range function to divide the date interval into 4 parts and return the A3 th two-members-sequence each time
4		=A1.select(B3(1)<=OrderDate && OrderDate<B3(2))	/Select records corresponding to the subinterval of each date
5		=file("Sales"+string(B3(1), "yyyyMMdd")+ ".txt").export@t(B4)	/Create file and export records selected by B4 to it

A3	Members	Members	Members	Members
	2014/01/20	2014/04/22	2014/07/22	2014/10/21
	2014/04/22	2014/07/22	2014/10/21	2015/01/20

THANKS

