SPL Base

esProc Tutorial

- JOINS

目录

CONTENTS

SPL

01 Objectified foreign key

1. Transform foreign key values to records in referenced table
2. Retain nonmatched records only

02 Normal joins

1. Using hashing algorithm
2. Using order-based merge
3. Foreign key joins
4. Cross product
5. A join with bin file

03 Position-based joins

1. Location by sequence numbers
2. Location by value positions
3. Location by field positions

04 A join with sequence

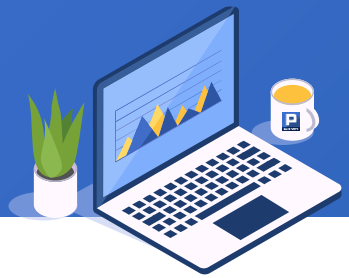
1. Left join
2. Degenerate to cross product
3. Complicated joins

CONTENTS

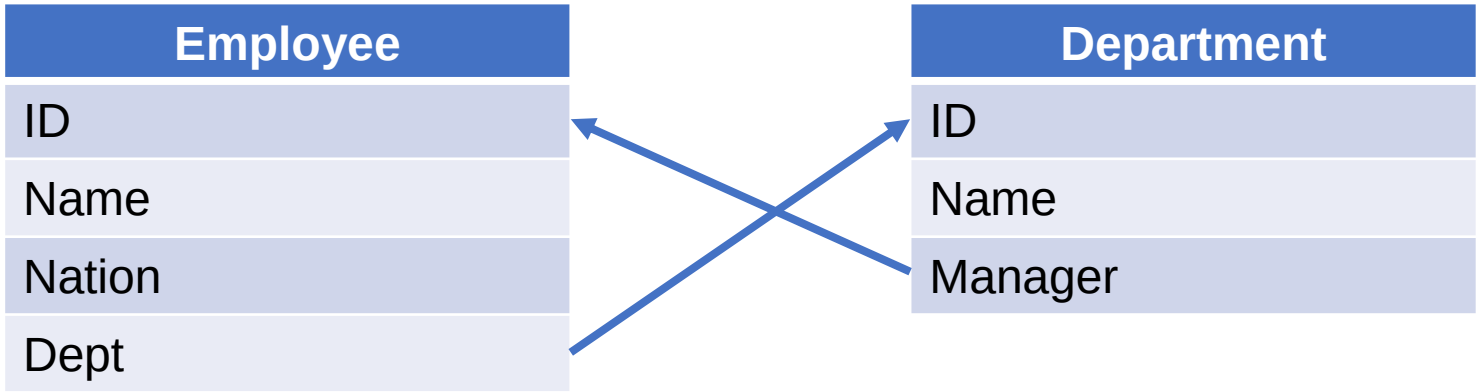
1. Transform foreign key values to records in referenced table
2. Retain nonmatched records only

Objectified foreign key

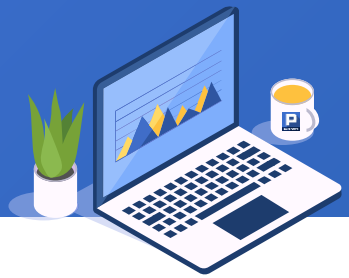
1. Transform foreign key values to records in referenced table



Query task: Find American employees under a Chinese manager according to *Employee* table and *Department* table.



1. Transform foreign key values to records in referenced table



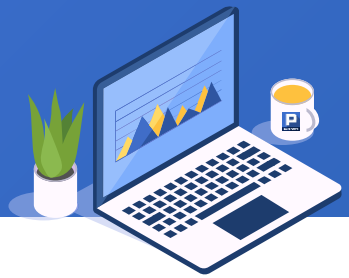
In SPL script below, **A.switch()** function transforms foreign key values into corresponding records in the referenced table:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Employee")	/Query *Employee* table
3	=A1.query("select * from Department")	/Query *Department* table
4	=A3.switch(Manager, A2:ID)	/switch function transforms *Department.Manager* into referenced records in *Employee* table
5	=A2.switch(Dept, A4:ID)	/switch function transforms Employee.Dept into referenced records in *Department* table
6	=A5.select(Nation=="American" && Dept.Manager.Nation=="Chinese")	/Get American employees whose managers are Chinese

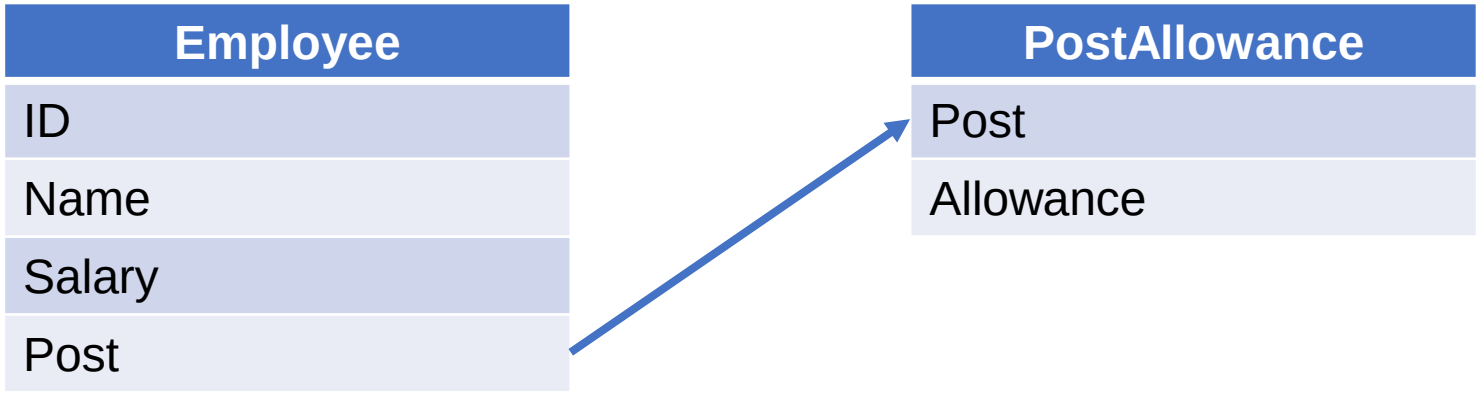
A6

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

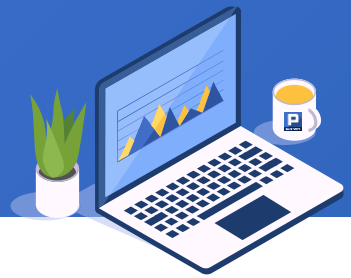
1. Transform foreign key values to records in referenced table



Query task: Calculate the total income for each employee (certain posts have allowances) according to *Employee* table and *PostAllowance* table.



1. Transform foreign key values to records in referenced table

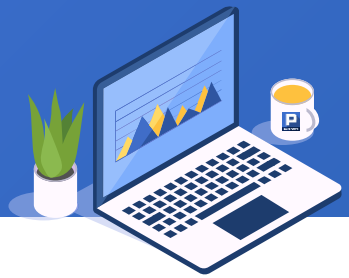


In SPL script below, **A.switch()** function transforms foreign key values into corresponding records in the referenced table; unmatched values will be set null:

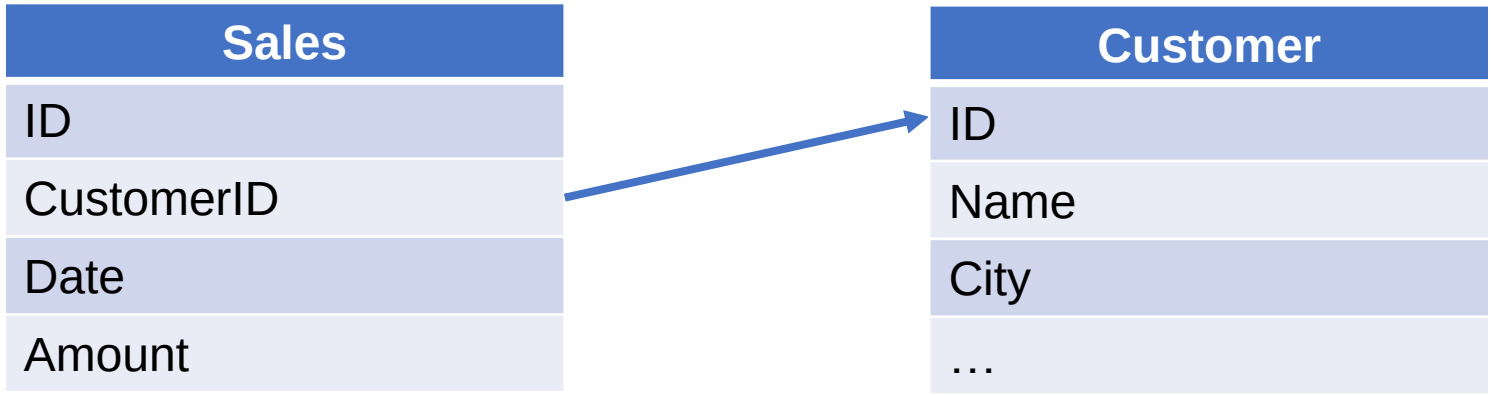
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Employee")	/Query *Employee* table
3	=A1.query("select * from PostAllowance")	/Query *PostAllowance* table
4	=A2.switch(Post, A3:Post)	/switch function transforms *Employee.Post* into referenced records in *PostAllowance* table; posts that haven’ t allowances are set nulls
5	=A4.new(ID,Name,Salary+Post.Allowance:Salary)	/Create a new table sequence and calculate salary for each employee

A5	ID	Name	Salary
	1	Rebecca	8000
	2	Ashley	12000
	…	…	…

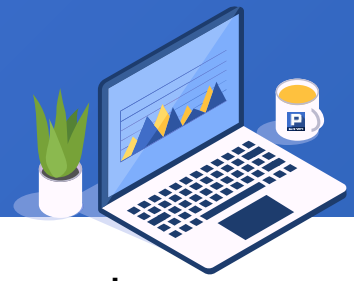
1. Transform foreign key values to records in referenced table



Query task: Calculate orders amount of each customer in Beijing in 2014 and sort result in descending order according to *Sales* table and *Customer* table.



1. Transform foreign key values to records in referenced table



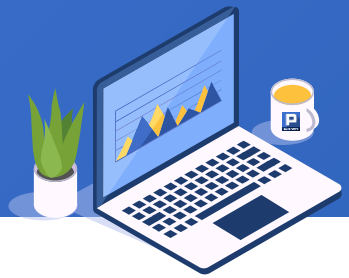
In SPL script below, **A.switch ()** function uses **@i** option to delete records from the referenced table where the foreign key values don’ t have matching values in the referencing table:

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

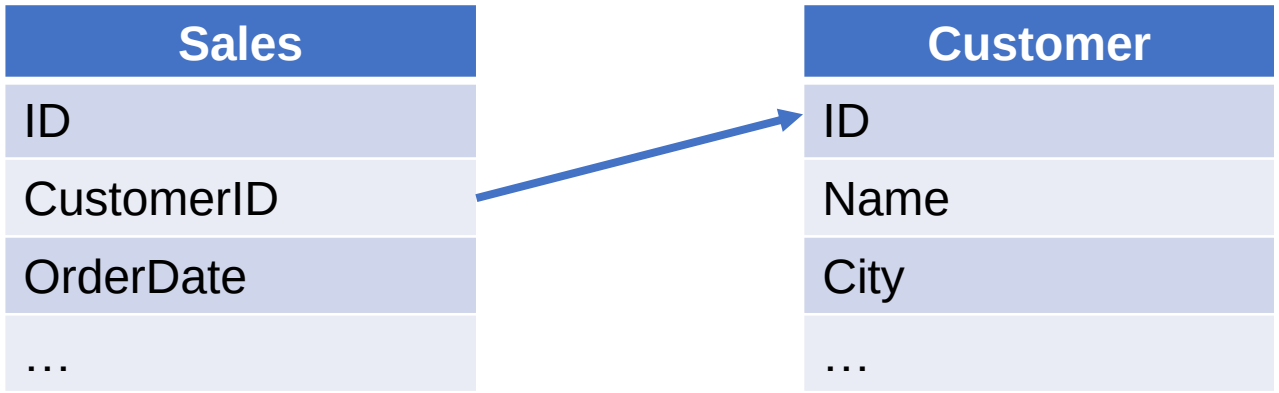
A5

Name	Amount
SAVEA	130672.64
HUN	23959.05
…	…

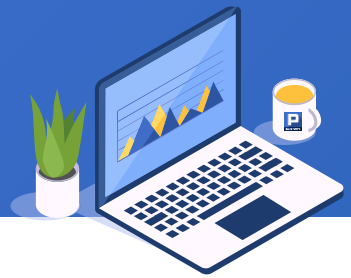
2. Retain nonmatched records only



Query task: Find new customers in 2014, which are those whose CustomerIDs are not included in *Customer* table according to *Sales* table and *Customer* table.



2. Retain nonmatched records only



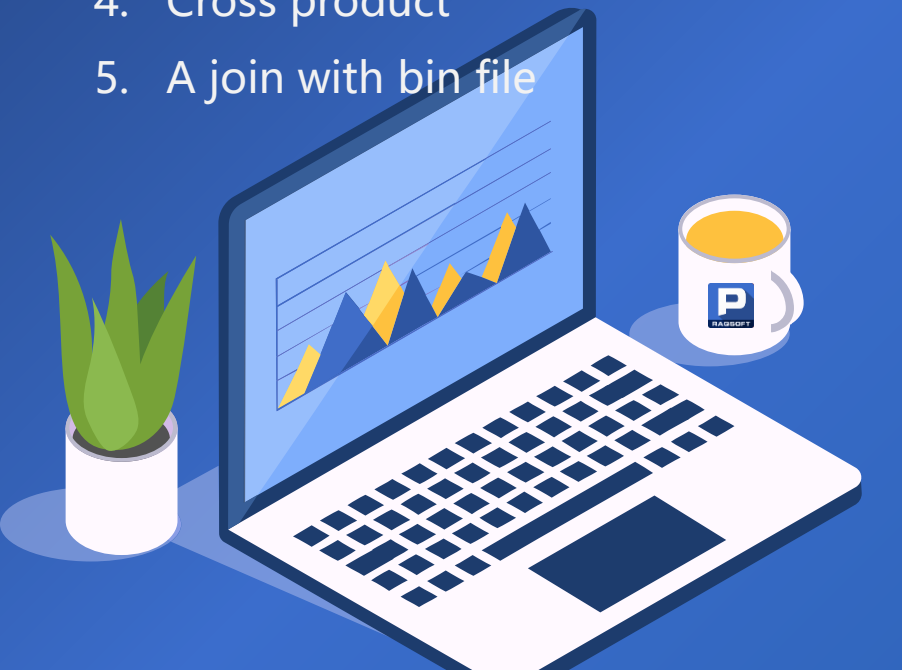
In SPL script below, **A.switch()** function works with **@d** option to retain only the nonmatched records; in this case the nonmatched foreign key values won't be set nulls:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Sales where year(OrderDate)=2014")	/Get records of the year 2014 from *Sales* table
3	=A1.query("select * from Customer")	/Query *Customer* table
4	=A2.switch@d(CustomerID ,A3:ID)	/@d option selects records from *Sales* table whose CustomerIDs don't exist in *Customer* table

A4

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

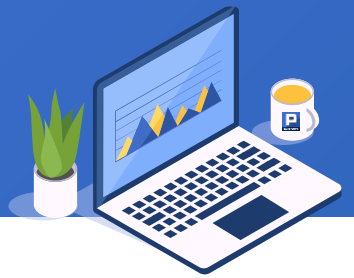
CONTENTS



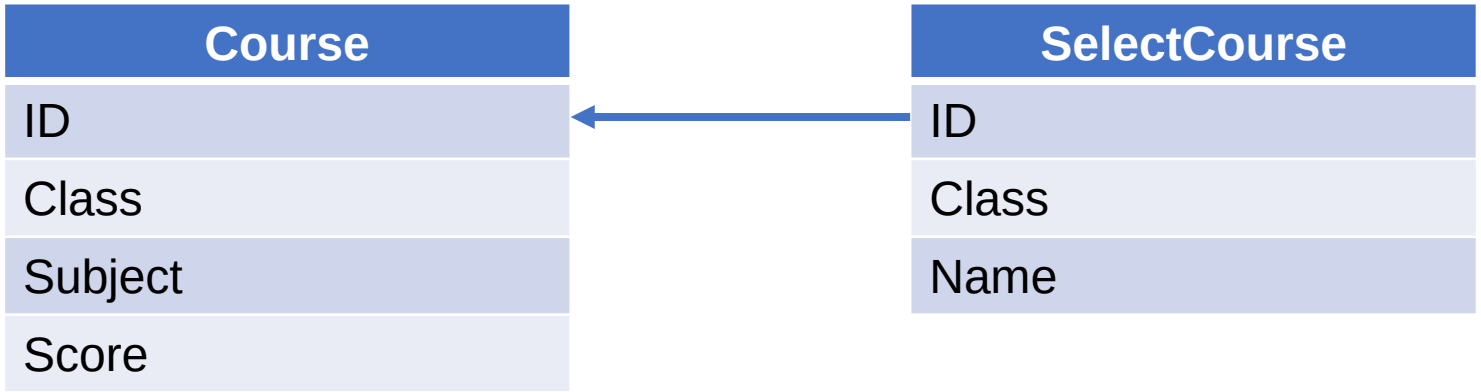
02

Normal joins

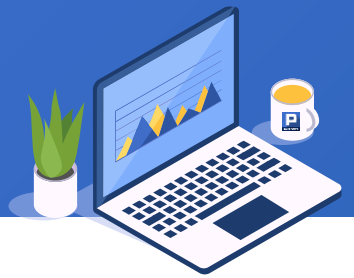
1. Using hashing algorithm



Query task: Calculate the number of students who select "Matlab" course according to *Course* table and *SelectCourse* table.



1. Using hashing algorithm

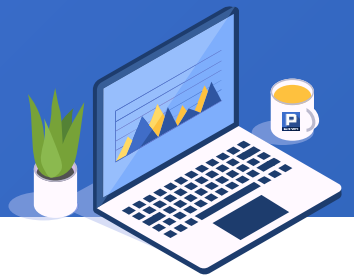


In SPL script below, **A.join()** function uses **@i** option to delete unmatched records from the right table:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Course")	/Query *Course* table
3	=A1.query("select * from SelectCourse")	/Query *SelectCourse* table
4	=A2.select(Name=="Matlab")	/Select the record with specified course
5	=A3.join@i(ID,A4:ID).count()	/Perform a filtering join using @i and calculate the number

A5	Value
	5

1. Using hashing algorithm

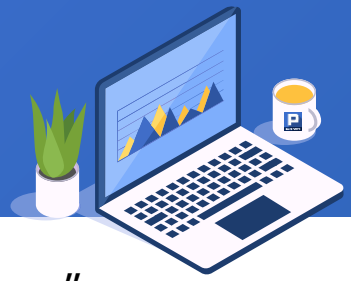


Query task: Find the total score of subjects for each student in class one according to *Score* table and *Student* table.

Score
ID
Subject
Score

Student
ID
Class
Name

1. Using hashing algorithm

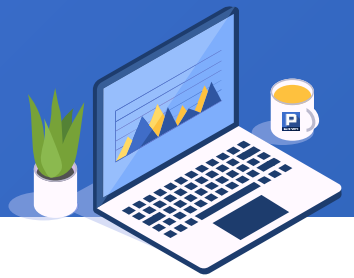


In SPL script below, we also use **@i** option to delete unmatched records. "Class one" is a constant condition by which the filtering join is performed.

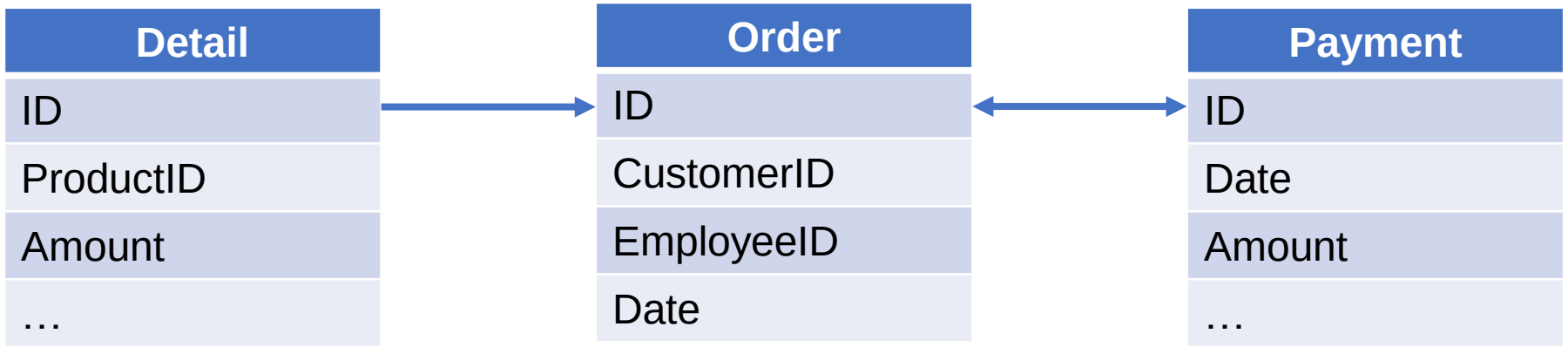
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Score")	/Query *Score* table
3	=A1.query("select * from Student")	/Query *Student* table
4	=A2.join@i(ID:"Class one", A3:ID:Class)	/Perform filtering join by *Student* table' s ID and Class using @i option
5	=A4.groups(ID; sum(Score):TotalScore)	/Group records to calculate each student' s total score

A5	ID	TotalScore
	1	230
	2	258
	3	228
	…	…

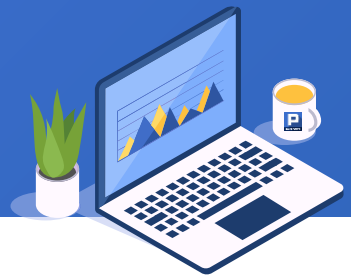
1. Using hashing algorithm



Query task: Find orders that payment hasn’ t done, that is, those where the paid amount is less than the payable amount, according to *Detail* table, *Order* table, and *Payment* table.



1. Using hashing algorithm

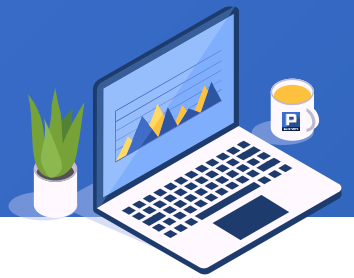


In SPL script below, **join()** function is used to do the join:

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

A9	ID	Amount	Pay
	AROUT	55492.0	35980
	BERGS	3398.55	1080
	…	…	…

2. Using order-based merge

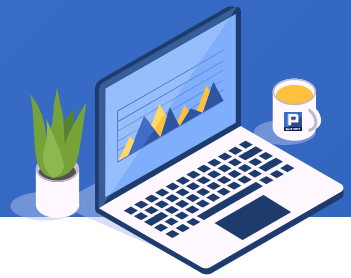


Query task: Find the sales for each customer in the year 2014 according to *Order* table and *Detail* table.

Order
ID
CustomerID
EmployeeID
Date

Detail
ID
ProductID
Amount

2. Using order-based merge

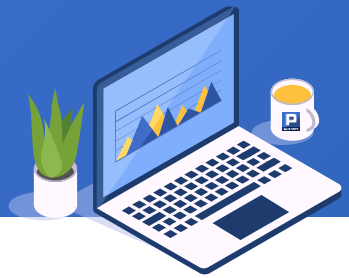


In SPL script below, **join()** function uses **@m** option to perform an order-based merge:

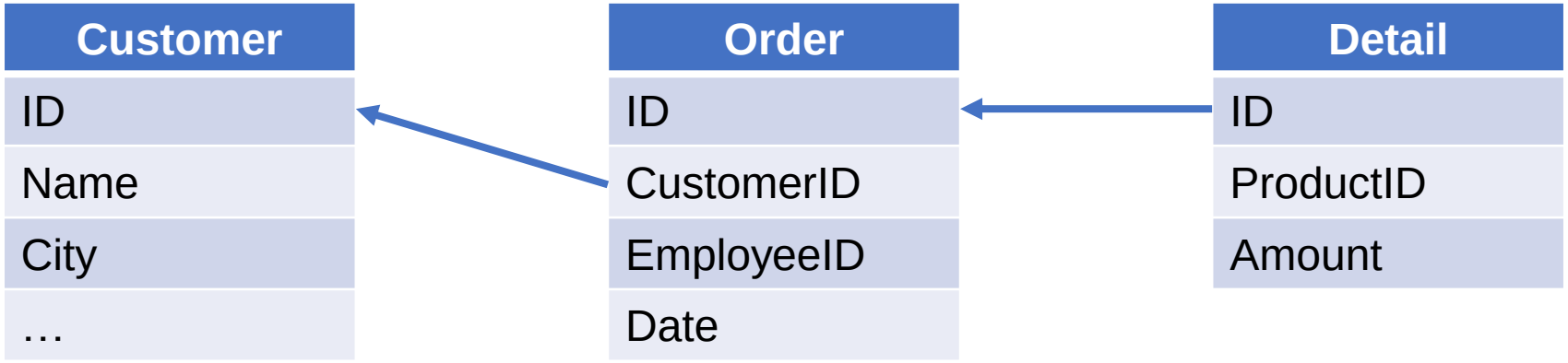
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Order where year(Date)=2014 order by ID")	/Select records of 2014 from *Order* table and sort them by ID
3	=A1.query("select * from Detail order by ID")	/Query *Detail* table and sort it by ID
4	=join@m(A2:Order,ID;A3:Detail,ID)	/join@m function merges the two tables by ordered ID
5	=A4.groups(Order.CustomerID:CustomerID; sum(Detail.Amount):Amount)	/Group records to sum each customer’ s sales amount

A5	CustomerID	Amount
	ALFKI	14848.0
	ANTON	4041.0
	…	…

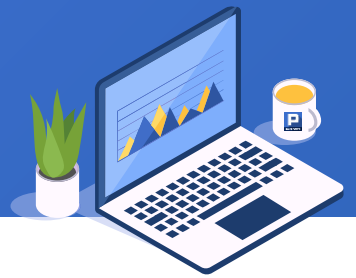
2. Using order-based merge



Query task: Find customers whose order amount is greater than ten thousand. As both Order table and Detail table can’ t be wholly loaded into the memory, we need to read them first as cursors and then perform order-based cursor merge.



2. Using order-based merge



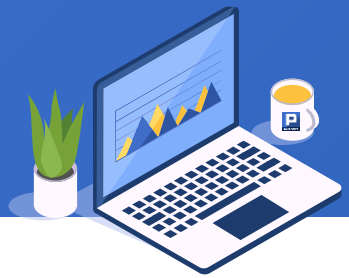
In SPL script below, we use **joinx()** function to do the order-based merge between cursors:

	A	B
1	=connect("db")	/Connect to database
2	=A1.cursor("select * from Order order by ID")	/Read *Order* table as a cursor ordered by ID
3	=A1.cursor("select * from Detail order by ID")	/Read *Detail* table as a cursor ordered by ID
4	=A1.query("select * from Customer")	/Query *Customer* table
5	=A2.switch@i(CustomerID,A4:ID)	/switch@i function replaces *Order.CustomerID* with referenced records in *Customer* table and delete unmatched records
6	=joinx(A5:Order,ID;A3:Detail,ID)	/joinx function merges *Order* table and *Detail* table by ordered ID
7	=A6.groups(Order.CustomerID.Name; sum(Detail.Amount):Amount).select(Amount>10000)	/Group records and calculate each customer’ s order amount and select those where the amount is greater than 10,000

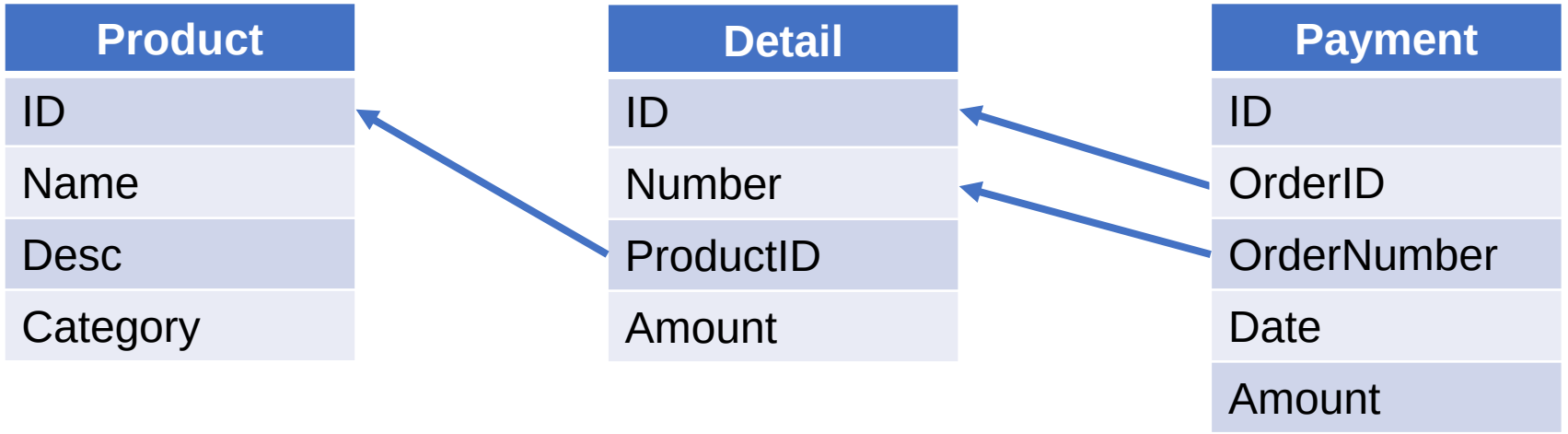
A7

Name	Amount
ALFKI	14848.0
AROUT	55492.0
…	…

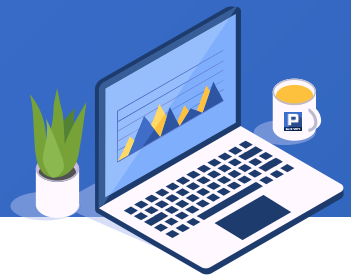
3. Foreign key joins



Query task: Find products for which money is paid and the total order amount is greater than 500 in 2014 according to *Product* table, *Detail* table and *Payment* table.



3. Foreign key joins

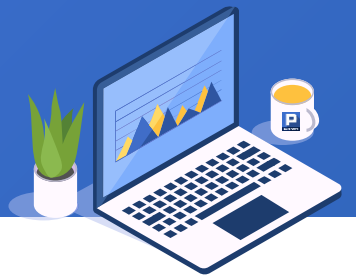


In SPL script below, we use A.join() function to perform a foreign key join with multi-primary-key:

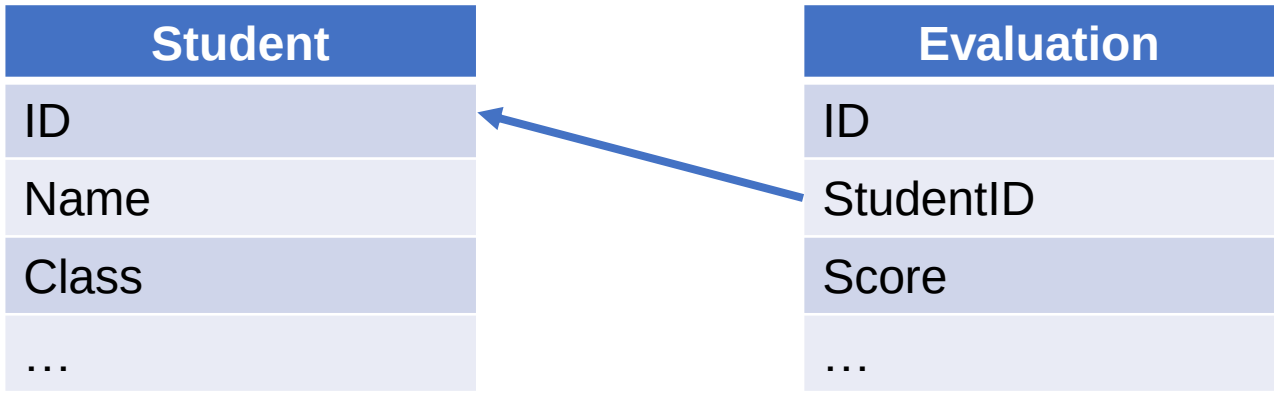
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Detail")	/Query *Detail* table
3	=A1.query("select * from Payment")	/Query *Payment* table
4	=A1.query("select * from Product")	/Query *Product* table
5	=A2.switch@i(ProductID,A4:ID)	/switch@i function transfers *Detail.ProductID* to referenced records in *Product* table
6	=A3.join(OrderID:OrderNumber,A5:ID:Number,~:Detail)	/A.join function relates *Detail* table and *Payment* table
7	=A6.select(year(Date)==2014 && Detail.Amount>500)	/Select desired records
8	=A7.new(ID,Date,Detail.Product.Name:Name,Detail.Amount:Amount)	/Create a new table sequence based on the selected records

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

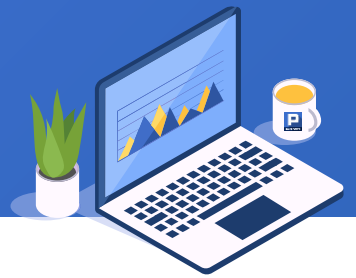
3. Foreign key joins



Query task: Find each student' s evaluation according to *Student* table and *Evaluation* table (There is a base score of 70).



3. Foreign key joins



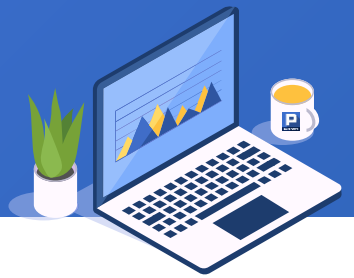
In SPL script below, **join()** function works with **@1** option to join the two tables according to the left table’ s structure:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Student")	/Query *Student* table
3	=A1.query("select * from Evaluation")	/Query *Evaluation* table
4	=A3.group(StudentID)	/Group *Evaluation* table by StudentID
5	=join@1(A2:Student,ID;A4:Evaluation,StudentID)	/join@1 function left joins *Student* table with grouped *Evaluation* table
6	=A5.new(Students.ID:ID,Student.Name:Name,70+Evaluation.sum(Score):Score)	/Create a new table sequence to calculate each student’ s total score, which is adding the evaluation score to the base score

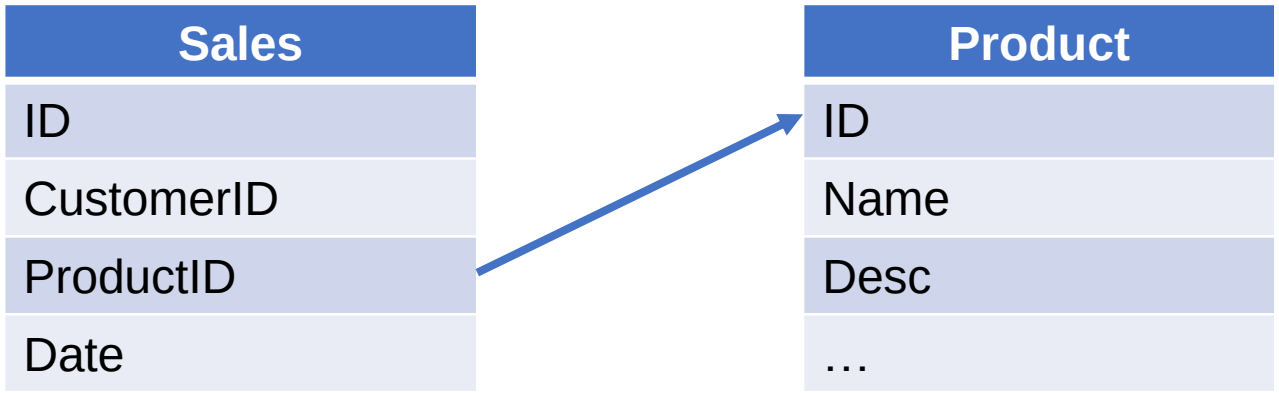
A6

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

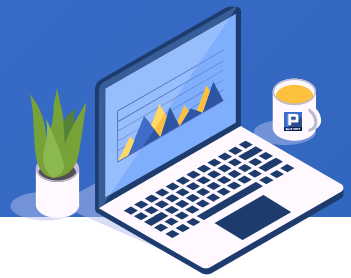
3. Foreign key joins



Query task: Compare each product’ s monthly sales amount in 2014 according to *Sales* table and *Product* table.



3. Foreign key joins

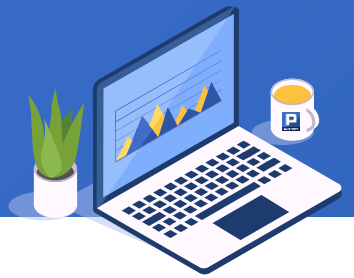


In SPL script below, **join()** function uses **@f** option to perform a full join:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select ProductID, month(Date) as Month from Sales where year(Date)=2014")	/product sales records of 2014 from *Sales* table
3	=A1.query("select * from Product")	/Query *Product* table
4	=A2.switch(ProductID ,A3:ID)	/switch function replaces *Sales.ProductID* with referenced records in *Product* table
5	=A4.group(Month)	/Group and sorts *Sales* table by Month
6	=A5.(~.group@1(ProductID).new(ProductID.Name:Product, count(~):Count))	/Group the grouped *Sales* table by removing duplicate products in each month and retain a product name field
7	=A6.("A6("+string(#)+"):"+string(#)+",Product").concat(";")	/Piece up parameter strings for join@f() funtion
A8	=join@f(${A7})	/join@f() performs a full join

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)
…	…	…	…	…	…	…	…	…	…	…	…

4. Cross product

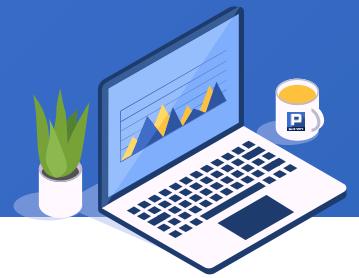


Query task: Find sandwiches having the most common ingredients according to *Sandwich* table and *Ingredient* table.

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
…	…

4. Cross product



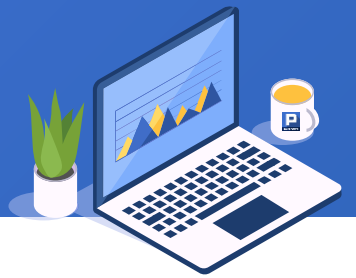
In SPL script below, we use **xjoin()** function to calculate cross product. Each record in the result set is calculated from records in two tables, rather than piecing together all field values.

	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 records from both the two tables
3	=A2.group@o(ID;Name,~.(Ingredient):Collection)	/group@o() performs merger grouping by ID and store all ingredients in Collection field
4	=xjoin(A3:A;A3:B,A.ID<ID)	/xjoin function calculates cross product on A3’s result and select sandwich couples of different IDs
5	=A4.new((A.Collection ^ B.Collection).len():Count, A.Name:Name1, B.Name:Name2).sort@z(Count)	/Find the number of common ingredients and sort records by duplicate count in descending order

A5

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

4. Cross product



Query task: Below is the data structure of a cross product table, which is the result of multiplying two matrices.

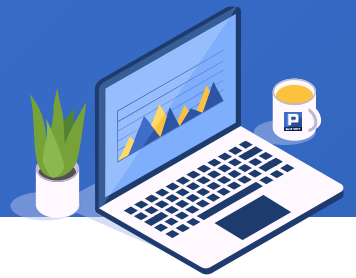
Matrix
row
col
value

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

The mathematical formula:

$$C = AB = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 1\times1+2\times2+3\times3 & 1\times4+2\times5+3\times6 \\ 4\times1+5\times2+6\times3 & 4\times4+5\times5+6\times6 \end{pmatrix} = \begin{pmatrix} 14 & 32 \\ 32 & 77 \end{pmatrix}$$

4. Cross product



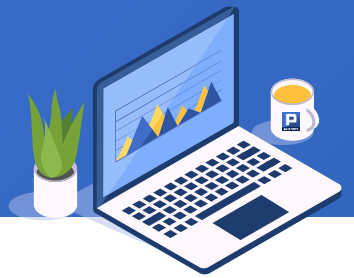
In the SPL script below, **xjoin()** function calculates cross product while performing a conditional filtering:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from MatrixA")	/Query MatrixA
3	=A1.query("select * from MatrixB")	/Query MatrixB
4	=xjoin(A2:A; A3:B, A2.col==A3.row)	/xjoin function calculates the two tables' cross product while filtering the result set by a condition
5	=A4.groups(A.row:row,B.col:col;sum(A.value * B.value):value)	/Group records and calculate every (row,column) value

A5

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

4. Cross product

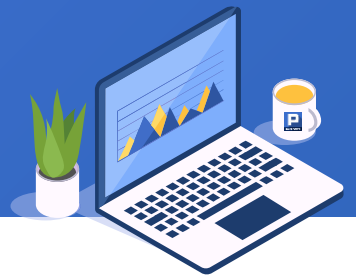


Query task: Get age groups among residents living in the community according to *Community* table and *Age* table.

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

4. Cross product

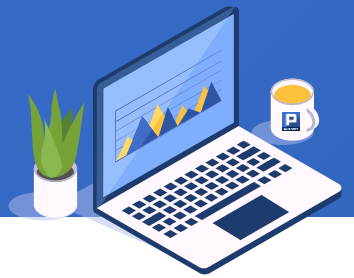


In SPL script below, **xjoin()@1** function performs a left join to calculate the cross product:

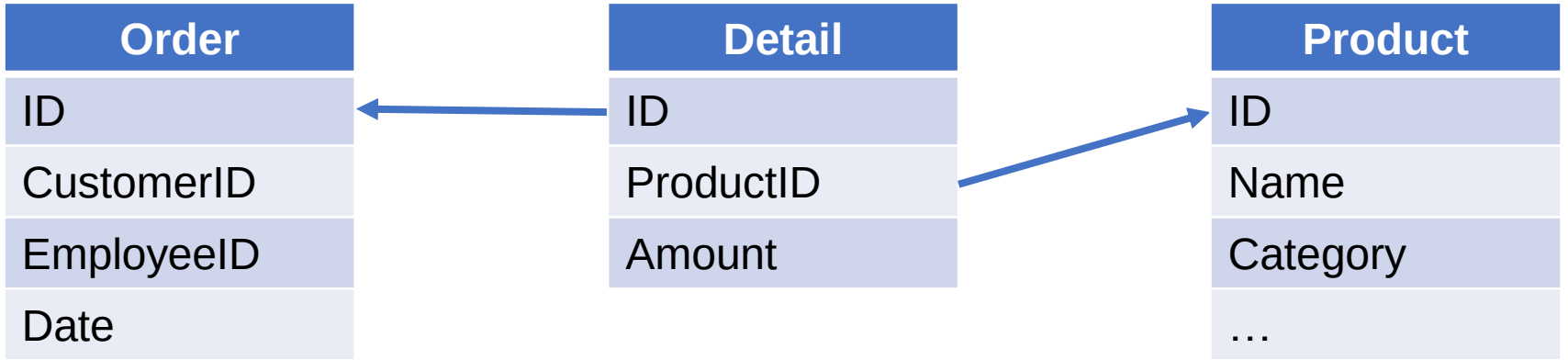
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Community")	/Query *Community* table
3	=A1.query("select * from Age")	/Query *Age* table
4	=xjoin@1(A2:Person; A3:Age, A3.Start<=Person.Age && A3.End>=Person.Age)	/xjoin@1() performs left join to get the cross product and selects records for each age group
5	=A4.new(Person.ID:ID, Person.Name:Name, Person.Age:Age,Age.Group:Group)	/Create a new table sequence to return the age group for each resident

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

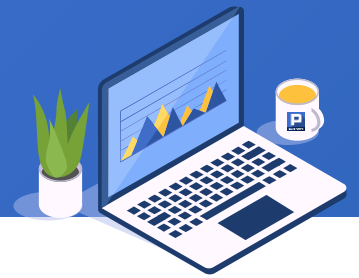
5. Joining files



Query task: Get how many pieces each product is sold in Jan., 2014. according to *Order* table, *Detail* table and *Product* table, which is a bin file ordered by ID.



5. Joining files



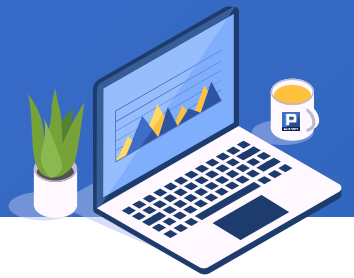
In SPL script below, we use **cs.joinx()** function to join the three files. The bin file must be ordered by the join field.

	A	B
1	=file("Detail.ctx").create().cursor()	/Create cursor for *Detail* table
2	=file("Order.ctx").create().cursor(;year(Date)==2014 && month(Date)==1)	/Create cursor for *Order* records of Jan., 2014
3	=file("Product.btx")	/Create a bin file object for *Product* table
4	=A1.joinx@i(ID,A2:ID)	/cs.joinx@i function performs a filtering join
5	=A4.joinx(ProductID,A3:ID,Name:ProductName)	/cs.joinx function joins *Detail* table and *Product* table by product’ s ID
6	=A5.groups(ProductName; count(~):Count)	/Group records and count the numbers each product is sold

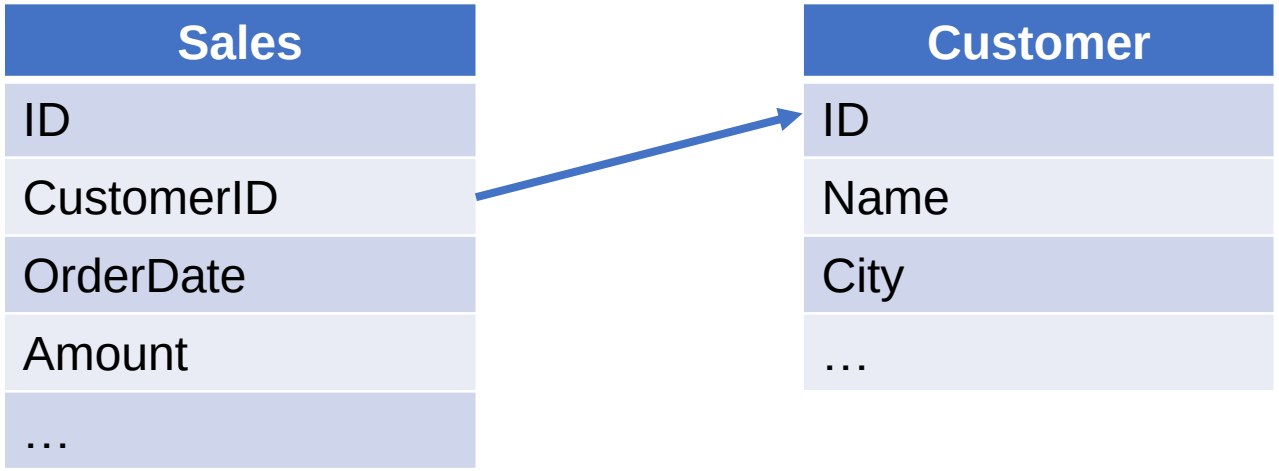
A6

ProductName	Count
Milk	32
Coffee	60
…	…

5. Joining files



Query task: Find customers who rank top 3 in 2014in terms of order amount according to *Sales* table and *Customer* table, which are bin files ordered by ID.



5. Joining files

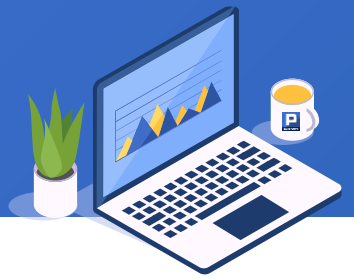
In SPL script below, we use **cs.joinx()** function to join with a segmentable bin file. @q function is used to speed up the computation as the bin file contain a relatively small volume of data.

	A	B
1	=file("Sales.btx").cursor@b().select(year(Date)==2014)	/Select *Sales* records of 2014 from the bin file and create cursor on them
2	=file("Customer.btx")	/Create a bin file object on *Customer* table that is ordered by customers' IDs
3	=A1.groups(CustomerID;sum(Amount):Amount)	/Group *Sales* records and calculate each customer's order amount
4	=A3.top(-3;Amount)	/Get top 3 customers in terms of order amount
5	=A4.joinx@q(CustomerID,A2:ID,Name:CustomerName).fetch()	/cs.joinx function joins Sales table and Customer table by customers' IDs; @q option is used to increase speed

A5

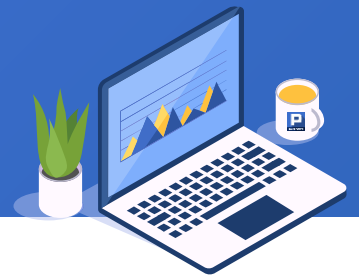
CustomerID	Amount	CustomerName
71	130672.64	SAVEA
63	64238.0	QUICK
20	53467.38	ERNSH

5. Joining files



Query task: Get the refund for each product in 2015 according to *Order* table, *Returns* table and *Product* table.

5. Joining files



In SPL script below, we use **cs.joinx()** function to join the files. Can use **@c** option to make computation faster if a cursor is ordered by the first join field. @c option can work with @q option:

	A	B
1	=file("Returns.btx").cursor@b().select(year(Date)==2015)	/Create cursor on *Returns* table
2	=file("Order.btx")	/Create bin file object on *Order* table
3	=file("Product.btx")	/Create bin file object on *Product* table
4	=A1.joinx@qc(OrderID,A2:ID,Amount;ProductID,A3:ID,Category)	/cs.joinx function joins *Returns* table with *Order* table by OrderID and with *Product* table By ProductID. Here @qc are used to speed up computation
5	=A4.groups(Category; sum(Amount))	/Group records by product and calculate the refund amount

A5

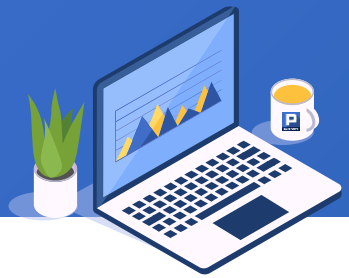
Category	Amount
Electric appliance	1854.5
Fruits	251.5
…	…

CONTENTS

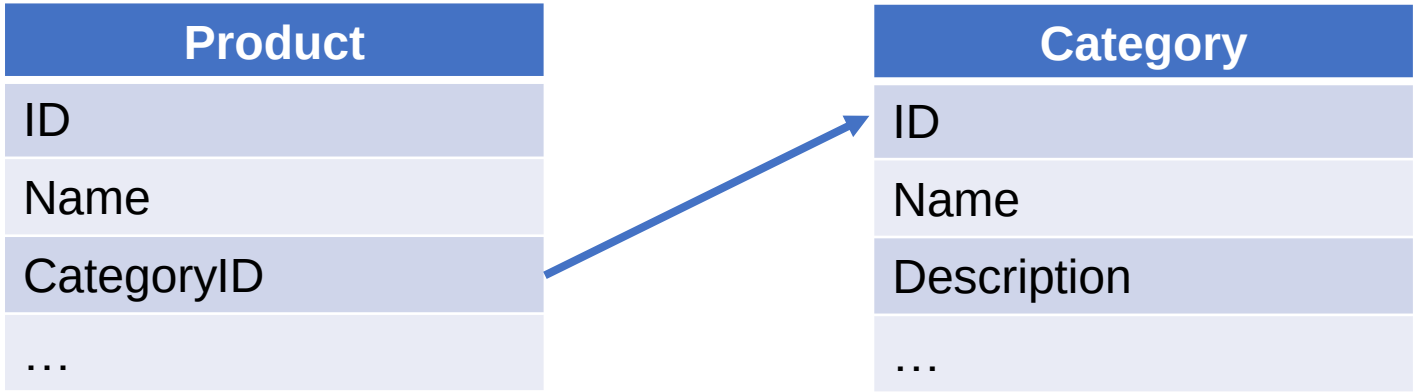
03

Position-based Joins

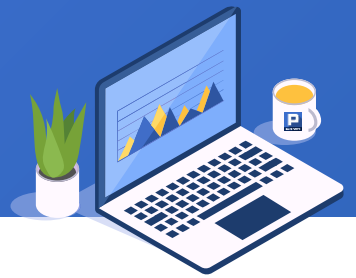
1. Location by sequence numbers



Query task: Find products under categories containing "drink" according to *Product* table and *Category* table:



1. Location by sequence numbers

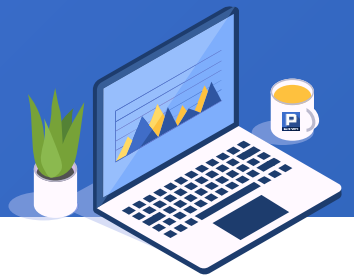


In SPL script below, **A.join()** function uses sequence-number-based location method, represented by symbol # to perform the join. #1 means the first field.

	A	B
1	=connect("demo")	/Connect to data source
2	=A1.query("select * from Product")	/Query *Product* table
3	=A1.query("select * from Category")	/Query *Category* table
4	=A3.select(like@c(Name, "*drink*"))	/Select records under categories whose name containing "drink" ; case-insensitive
5	=A2.join@i(CategoryID,A4:#1)	/Perform filtering join through *Product.CategoryID* and the first field (ID) in *Category* table

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

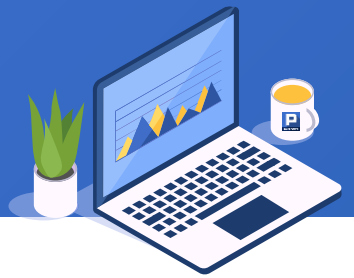
2. Location by value positions



Query task: Shuffle values of a certain column in a database table and write new values back. Below is *REF_VALUES* table:

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

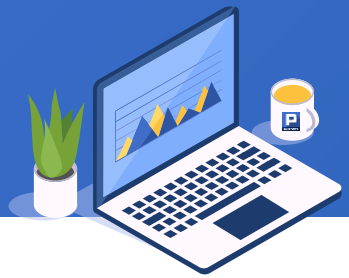
2. Location by value positions



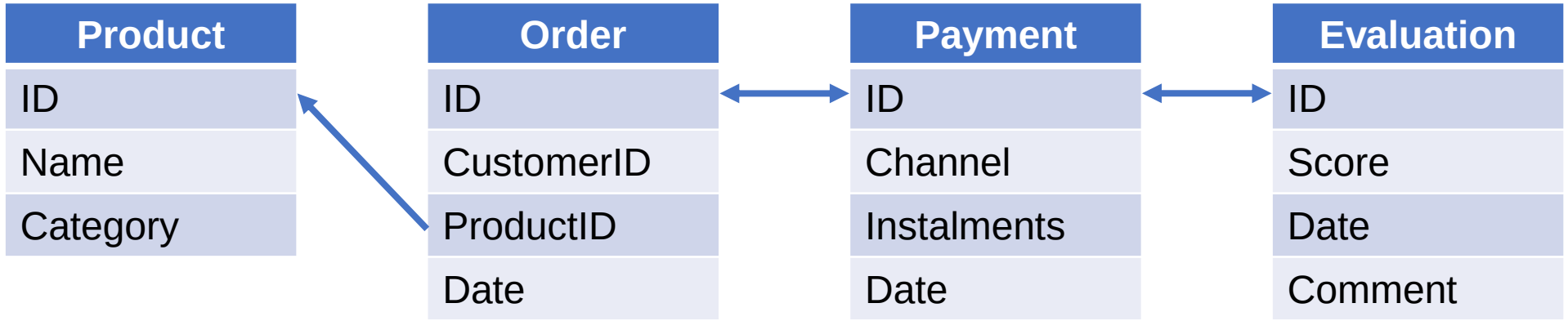
In SPL script below, **join()** function **@p** option to perform a join according to positions:

	A	B
1	=connect("demo")	/Connect to database
2	=A1.query("select ID,ORIGINAL_VALUE from REF_VALUES")	/Query *REF_VALUES* table
3	=A2.sort(rand())	/Sort selected fields randomly to rearrange values
4	=join@p(A2.(ID);A3.(ORIGINAL_VALUE))	/join@p joins the original IDs and the shuffled values by positions
5	=A1.update@u(A4, REF_VALUES, ID:_1, SHUFFLED_VALUE:_2; ID)	/Update shuffled values to database table *REF_VALUES* by primary key ID

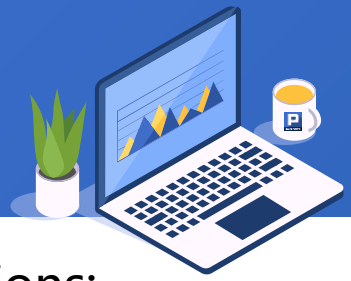
2. Location by value positions



Query task: Find orders in 2014 that don’ t use installment loan and calculate average evaluation of each product category. *Order* table, *Payment* table and *Evaluation* table have same order ID values. Relationships between involved tables are as follows:



2. Location by value positions

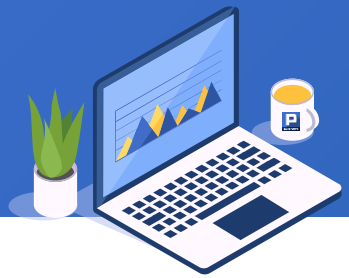


In the SPL script below**, join()** function uses **@p** option to perform a join by positions:

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

A9	Category	Score
	Electric appliance	3.98
	Fruits	3.86
	…	…

3. Location by field positions



The table records populations of cities around the world:

Continent	Country	City	Population
Africa	Egypt	Cairo	6789479
Asia	China	Shanghai	24240000
Europe	Britain	London	7285000

List European and African cities having more than 2 million population and the numbers in two column groups, with each group ordered by population in descending order. The expected result:

Europe City	Population	Africa City	Population
Moscow	8389200	Cairo	6789479
London	7285000	Kinshasa	5064000
St Petersburg	4694000	Alexandria	3328196

3. Location by field positions

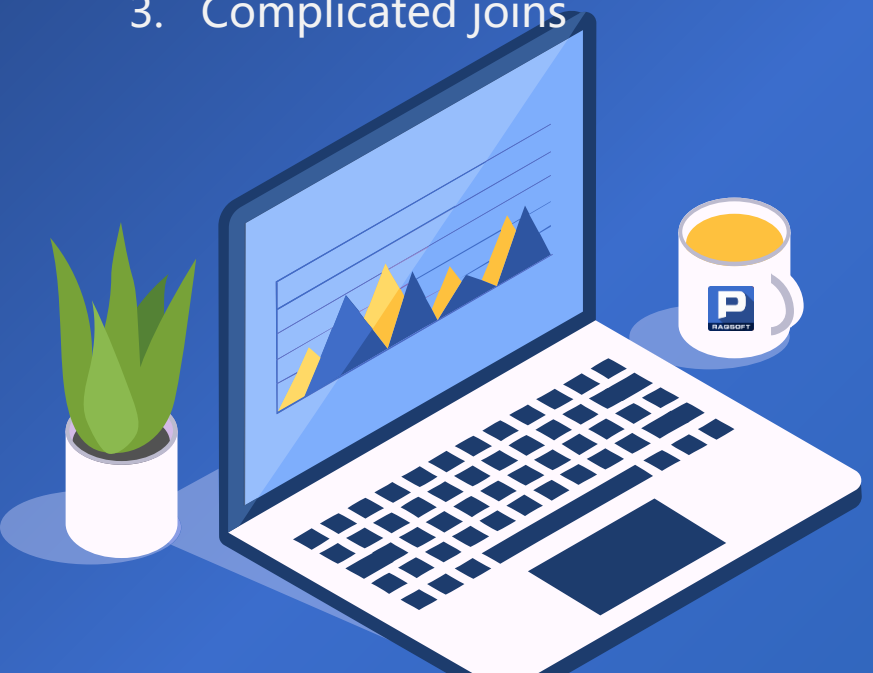
In SPL script below, **A.paste()** function relates Europe records to original Africa records with field names omitted and add values to the new table sequence by the order of fields

	A	B
1	=connect("db").query("select * from World where Continent in('Europe', 'Africa') and Population >= 2000000")	/Connect to database to get population records of European and African cities having more than 2 million population
2	=A1.select(Continent:"Europe")	/select() function gets records of European cities
3	=A1.select(Continent:"Africa")	/select() function gets records of African cities
4	=A2.new(City:'Europe City',Population:'Europe Population','Africa City','Africa Population')	/Create a new table sequence based on European data
5	=A4.paste(A3.(City):#3,A3.(Population):#4;1)	/paste() function relates A4’s Europe records to A3’s original African records by field records to get desired value sequences to fill into the 3rd column and 4th column

A5

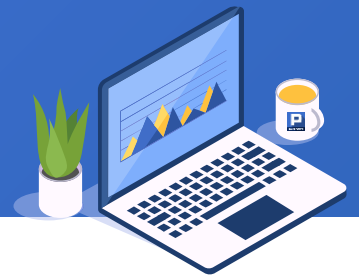
Europe City	Population	Africa City	Population
Moscow	8389200	Cairo	6789479
London	7285000	Kinshasa	5064000
St Petersburg	4694000	Alexandria	3328196
…	…	…	…

CONTENTS

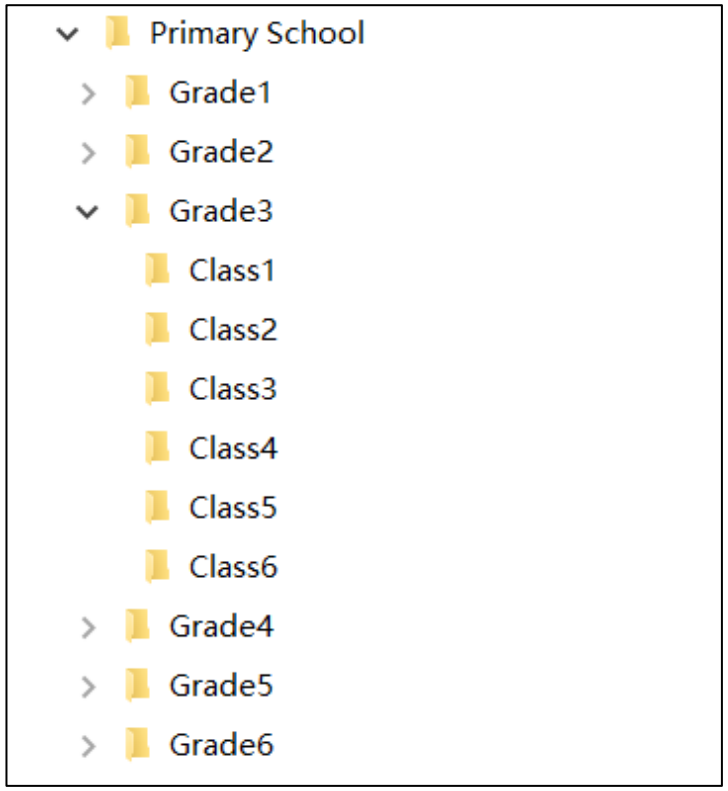


04 A join with sequence

1. Left join

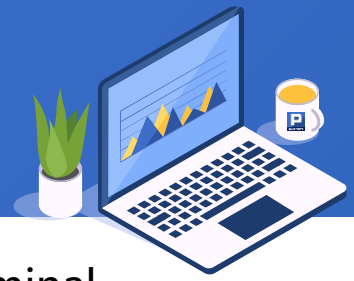


Query task: Calculate the ratio of each terminal type used by students in a primary school during online learning. Below is the directory holding questionnaire files of all classes:



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
…	…	…

1. Left join



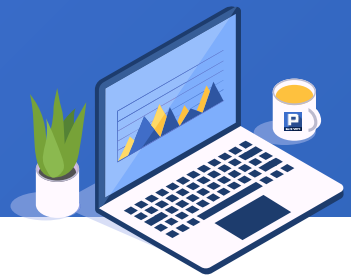
In SPL script below, **A.news()** function relates the table sequence to the sequence of split terminal names.

	A	B	
1	=directory@ps("D:/Primary School")		/Traverse the directory iteratively to list all files
2	for A1	=file(A2).xlsimport@t()	/Load into the questionnaire Excel files iteratively
3		=@+=B2.len()	/Calculate the total rows, which is the total number of students
4		=B2.news(B2.TERMINAL.split@c( ); ID, STUDENT_NAME, ~:TERMINAL)	/news function relates questionnaire files to split terminal names
5		=B4.groups(TERMINAL; count(~):Count)|@	/Group and count the number of each terminal and add the result to this cell if B4’ s union result exceeds the memory space,
6	=B5.groups(TERMINAL;string(sum(Count)/B3, "#.##%"):PERCENTAGE)		/Group and summarize the aggregate result of each class

A6

TERMINAL	PERCENTAGE
PC	70%
Pad	56.67%
Phone	93.33%

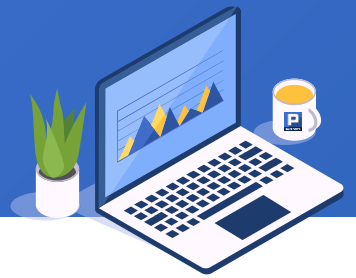
1. Left join



Query task: Find the most frequently used labels for each author according to *PostRecord* table.

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
…	…	…	…

1. Left join



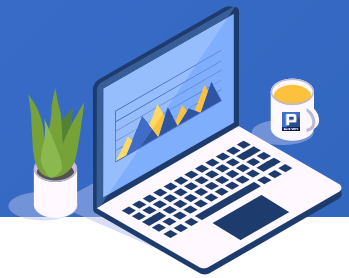
In SPL script below, **A.news()** function uses **@1** option to do a left join:

	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)	/A.news@1 function left joins the sequence of labels and retain the post records even it hasn’ t a label
4	=A3.groups(Author,Label;count(~):Count)	/Group and count labels used by every author
5	=A4.group(Author).conj(~.maxp@a(Count))	/Group by authors to get the most frequently used labels and retain both if there is a tie

A5

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

2. Degenerate cross product

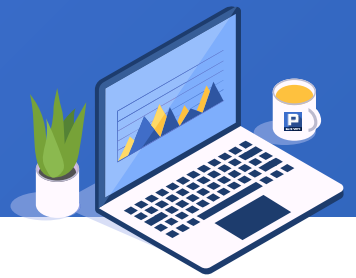


Query task: Find the available teachers for each course according to the *Teachers* file and *Courses* file.

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
…	…

2. Degenerate to cross product



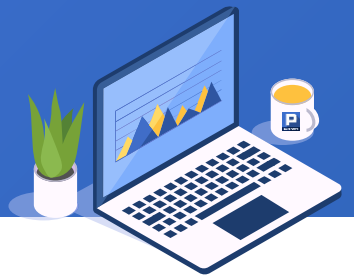
In SPL script below, **A.news()** function calculates cross product of a table and a sequence:

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

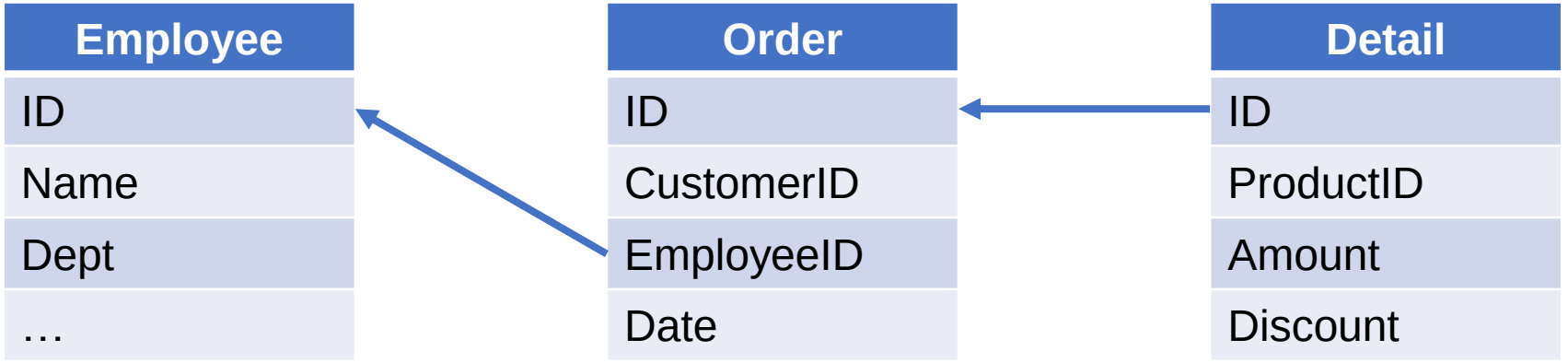
A5

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

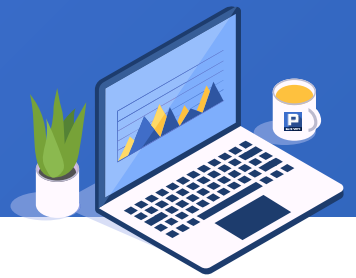
3. Complicated joins



Query task: A company has the policy that salespeople whose single order amount is greater than 1000 will be given a 5% performance award . Then, what is the actual sale amount of each salesperson?



3. Complicated joins



In SPL script below, **A.news()** function performs the join and the specified calculation:

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

A8	Name	Amount
	Alexis	358882.02
	Emily	432435.85
	…	…

THANKS

for your attention