How to implement in, exists, any and all

CONTENTS

SPL

01 IN

1. Set of constants
2. Non associated subquery returns primary key
3. Non associated subquery returns non primary key
4. Large amount of data returned by subquery
5. Associated subquery

02 EXISTS

1. Subquery associates primary key
2. Subquery associates non primary key
3. Large amount of data returned by subquery
4. Subquery non equivalent Association

03 NOT IN, NOT EXISTS

1. NOT IN
2. NOT EXISTS
3. Double negative

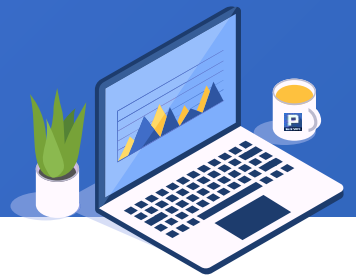
04 ANY, ALL

1. ANY(or SOME)
2. ALL

CONTENTS

01 IN 1

1. Set of constants



Take the employee table as an example to count the average salaries of all departments 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
…	…	…	…

1. Set of constants

The SQL statement is as follows:

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

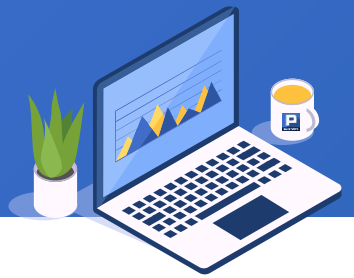
1. Set of constants

When there are no more than 9 constant items, you can use A.contain() function to filter. SPL is as follows:

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

A4	DEPT	SALARY
	Finance	7833.33
	HR	7187.5
	Marketing	7977.27
	…	…

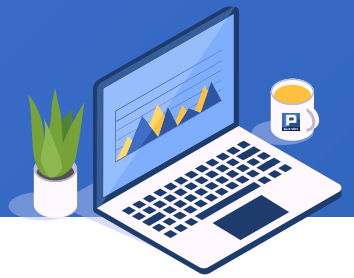
✦ 1. Set of constants



Query the total monthly sales of key customers ["SAVEA","QUICK", ...] in 2014. The sales table is as follows:

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
…	…	…	…	…

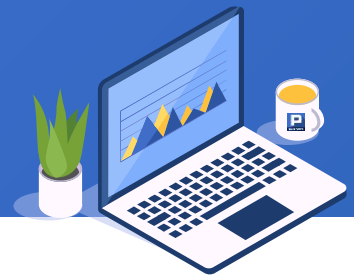
1. Set of constants



The SQL statement is as follows:

```
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
```

1. Set of constants

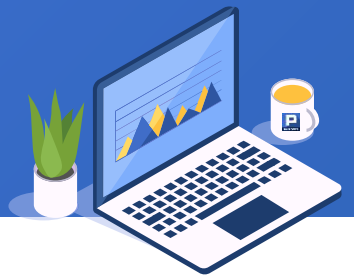


When there are more than 10 constant items, you can sort the constant sequence, and then use the @b option of A.contain() function to perform binary search. SPL is as follows:

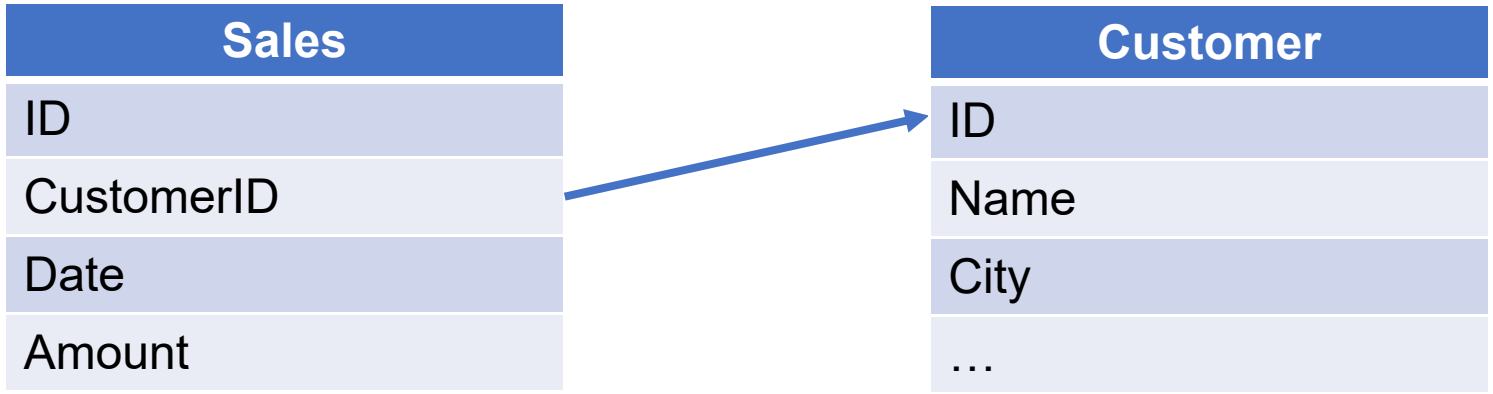
	A	B
1	=connect("db").query("select * from Sales")	/Connect to database and query employee table
2	=["SAVEA","QUICK","ERNSH","HUN","RATTC","HANAR","FOLKO","QUEEN,MEREP","WHITC","FRANK","KOENE"].sort()	/Create a constant sequence of key customers and sort them
3	=A1.select(year(Date)==2014 && A2.contain@b(Customer))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

A4	Month	Amount
	1	16947.3
	2	27793.3
	3	14602.7
	…	…

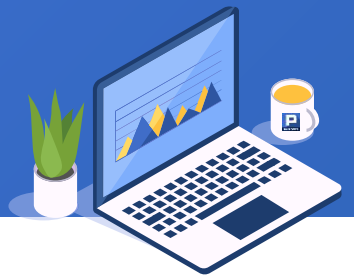
2. Non associated subquery returns primary key



Query the sales of customers in Beijing in 2014, in descending order. The sales and customer tables are as follows:



2. Non associated subquery returns primary key



The SQL statement is as follows:

```
select
        CustomerID, sum(Amount) as Amount
from
        Sales
where
        year(Date)=2014
         and CustomerID in ( select ID from Customer where City='Beijing' )
group by CustomerID
order by Amount desc
```

✦ 2. Non associated subquery returns primary key

Used the @i option of A.switch() function. Delete the record when no foreign key mapping is found. SPL is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Sales where year(Date)=2014")	/Query 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 the @i option of the switch function to keep only the records of Beijing customers
5	=A4.groups(CustomerID.Name:Name; sum(Amount):Amount).sort@z(Amount)	/Group and summarize the sales of each customer and arrange them in descending order of total sales

A5	Name	Amount
	SAVEA	130672.64
	HUN	23959.05
	…	…

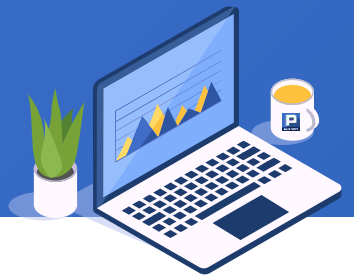
2. Non associated subquery returns primary key

Check how many students in each class have taken "matlab" course. The course table and course selection table are as follows:

Course
ID
Subject
Score

SelectCourse
ID
Class
Name

2. Non associated subquery returns primary key



The SQL statement is as follows:

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

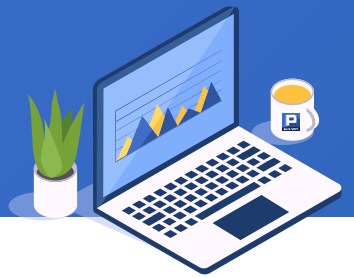
✦ 2. Non associated subquery returns primary key

The @i option of the A.join() function is used here, deleting mismatched records. SPL is as follows:

	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 specific course from the course table
5	=A3.join@i(ID, A4:ID)	/Use @i option of join function for join filtering
6	=A5.groups(Class; count(1):SelectCount)	/Group and summarize the number of people registered in each class

A6	Class	SelectCount
	Class 1	3
	Class 2	5
	…	…

3. Non associated subquery returns non primary key

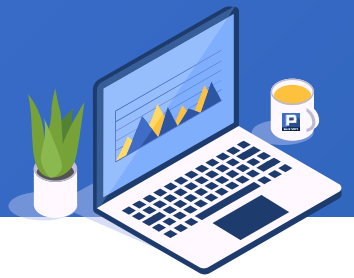


Check the number of students in each class who have a subject score of more than 80. The score table and student table are as follows:

Score
StudentID
Subject
Score

Student
ID
Class
Name

3. Non associated subquery returns non primary key



The SQL statement is as follows:

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

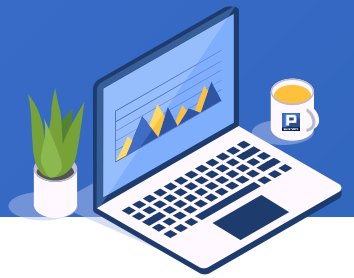
✦ 3. Non associated subquery returns non primary key

After the subquery is filtered, it is read into memory after deduplication by the joined field, and then it becomes a case similar to the primary key. SPL is as follows:

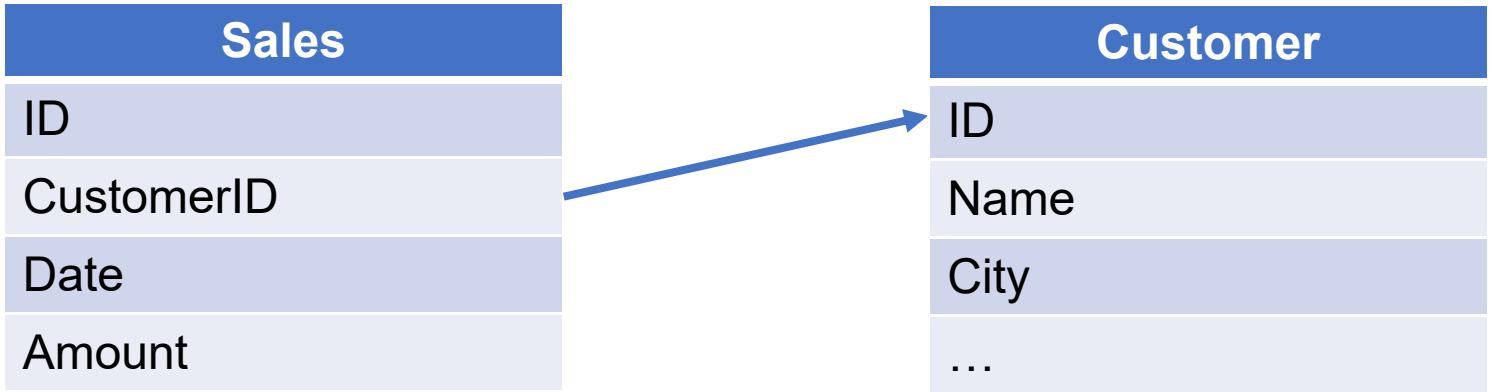
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Student")	/Query student table
3	=A1.query("select * from Score")	/Query score table
4	=A3.select(Score>80)	/Select records with score over 80
5	=A4.id(StudentID)	/Use ID function to deduplicate by Student ID
6	=A2.join@i(ID, A5)	/Use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

A7	Class	StudentCount
	Class 1	9
	Class 2	11
	…	…

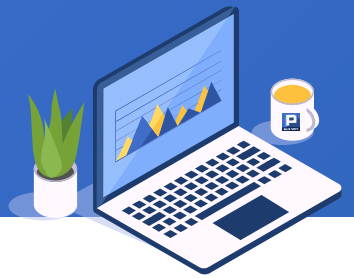
3. Non associated subquery returns non primary key



Query the number of customers with sales records in each city in 2014. The sales and customer tables are as follows:



3. Non associated subquery returns non primary key



The SQL statement is as follows:

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

3. Non associated subquery returns non primary key

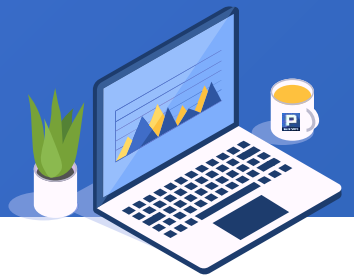
You can use the @o option of the groups function to speed up when the data is in order by the deduplication field.

SPL is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Customer")	/Query customer table
3	=A1.query("select * from Sales where year(Date)=2014 order by CustomerID")	/Query 2014 sales records and sort by customer ID
4	=A3.groups@o(ID)	/Use the groups function to deduplicate by customer ID, and use the @o option when data is in order
5	=A2.join@i(ID, A4:CustomerID)	/Use A.join@i() function to join and filter
6	=A5.groups(City; count(1):CustomerCount)	/Group and summarize the number of customers in each city

A6	City	CustomerCount
	Dongying	6
	Tangshan	7
	…	…

4. Large amount of data returned by subquery



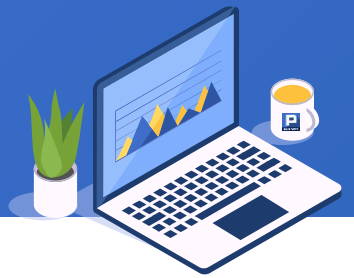
When order details and order table are too large to be loaded into memory, ordered merging by cursor is needed.

Query the quantity of orders signed by each salesperson whose actual sales amount exceeds 1000, in descending order.

Order
ID
CustomerID
EmployeeID
Date

Detail
ID
ProductID
Amount
Discount

4. Large amount of data returned by subquery



The SQL statement is as follows:

```
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
```

4. Large amount of data returned by subquery

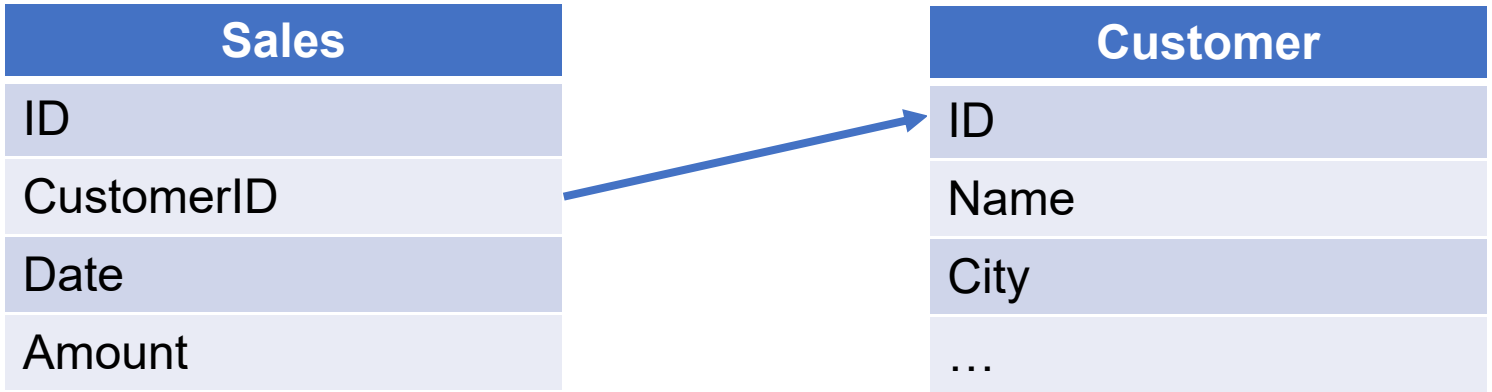
The joinx() function is used here for orderly merging. SPL is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.cursor("select * from Order order by ID")	/Create order table cursor
3	=A1.cursor("select * from Detail order by ID")	/Create order detail table cursor
4	=A3.select(Amount*(1-Discount)>1000)	/Select records with actual sales greater than 1000
5	=joinx(A2:Order,ID;A4:Detail,ID)	/Using joinx function to merge the cursors of order table and order detail table in order
6	=A5.groups(Order.EmployeeID:EmployeeID; icount(Order.ID):OrderCount).sort@z(OrderCount)	/Group and summarize the order quantity of each sales person, and arrange them in descending order according to the order quantity

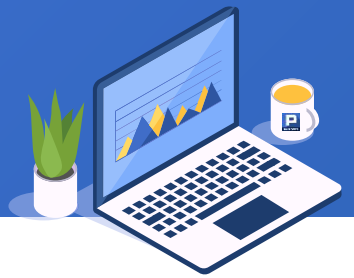
A6	EmployeeID	OrderCount
	2	3
	3	3
	…	…

5. Associated subquery

When there is no null value, the in with subquery can be described by exists. The example in Section 3 is as follows: query the number of customers with sales records in each city in 2014:



5. Associated subquery



The SQL statement is as follows, which is transformed into exists associated subquery:

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

The solution of SPL is the same for in subquery and exists subquery of the same requirement.

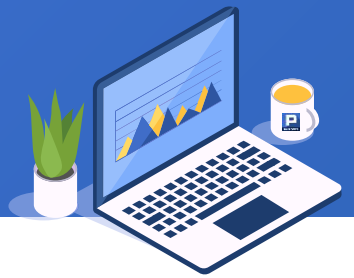
CONTENTS

1. Subquery associates primary key
2. Subquery associates non primary key
3. Large amount of data returned by subquery
4. Subquery non equivalent Association

02

EXISTS

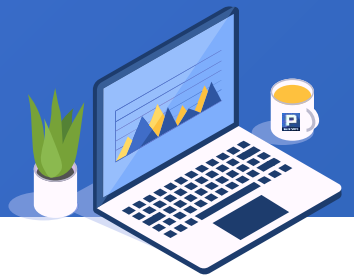
1. Subquery associates primary key



Check the average score of each boy in class one. The score table and student table are as follows:

Score		Student
StudentID	→	ID
Class	→	Class
Subject		Name
Score		Gender

1. Subquery associates primary key



The SQL statement is as follows:

```
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
```

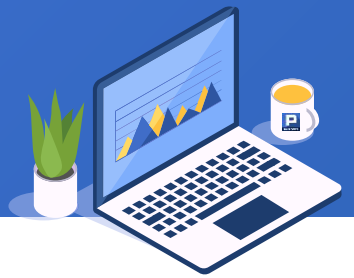
1. Subquery associates primary key

The idea of this chapter is the same as in subquery. In fact, this exists can also be written with in. SPL is as follows:

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

A6	StudentID	Score
	1	76
	3	74
	…	…

2. Subquery associates non primary key

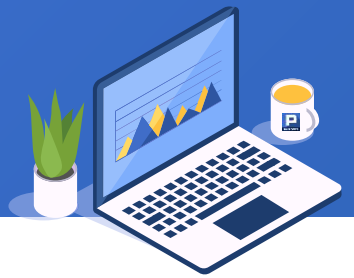


Query the quantity of various products with sales over 500 in 2014. The sales table and product table are as follows:

Sales
ID
ProductID
Date
Amount

Product
ID
Name
Category
…

2. Subquery associates non primary key



The SQL statement is as follows:

```
select
        Category, count(*) as ProductCount
from
        Product
where
        exists (
                select * from Sales
                where year(Date)=2014 and Amount>500
                        and Product.ID=Sales.ProductID
        )
group by Category
```

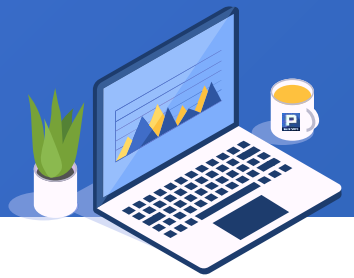
2. Subquery associates non primary key

SPL is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Product")	/Query product table
3	=A1.query("select * from Sales")	/Query sales table
4	=A3.select(year(Date)==2014 && Amount>500)	/Select orders with sales over 500 in 2014
5	=A3.id(ProductID)	/Deduplication by product ID
6	=A2.join@i(ID, A5)	/Use A.join@i() function to join and filter
7	=A6.groups(Category; count(1):ProductCount)	/Group and summarize the product sales quantity of each category

A7	Category	ProductCount
	Electric appliance	2
	Fruits	5
	…	…

3. Large amount of data returned by subquery

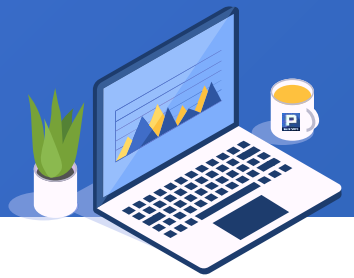


When the exist subquery returns a large amount of data, similar to the in subquery, it can also be solved by the orderly merging of cursors. Query the number of orders that did not use discounts per month in 2014.

Order
ID
CustomerID
EmployeeID
Date

Detail
ID
ProductID
Amount
Discount

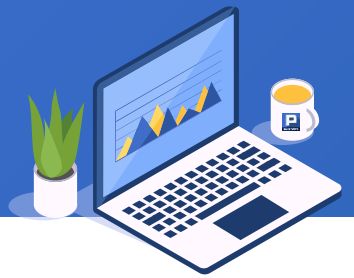
3. Large amount of data returned by subquery



The SQL statement is as follows:

```
select
        month(Date) as Month, count(*) as OrderCount
from
        Order
where
        exists ( select * from Detail where Discount = 0 and Order.ID = Detail.ID )
group by Month
order by Month
```

3. Large amount of data returned by subquery



The joinx() function is used here for orderly merging. SPL is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.cursor("select * from Order where year(Date)=2014 order by ID")	/Create order table cursor and select 2014 records
3	=A1.cursor("select * from Detail order by ID")	/Create order detail table cursor
4	=A3.select(Discount==0)	/Select records that do not use discounts
5	=joinx(A2:Order,ID;A4:Detail,ID)	/Using joinx function to merge the cursors of order table and order detail table in order
6	=A5.groups(month(Order.Date):Month; icount(Order.ID):OrderCount)	/Group and summarize the order quantity of each month

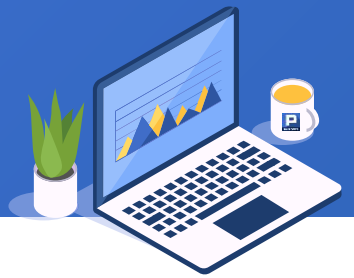
A6	Month	OrderCount
	1	16
	2	25
	…	…

4. Subquery non equivalent Association

Query the sales amount of orders with a time span of more than one year. The order details table is as follows:

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
…	…	…	…	…

4. Subquery non equivalent Association



The SQL statement is as follows:

```
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
```

4. Subquery non equivalent Association

SPL 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 date
4	=A3.select(interval(~.min(DELIVERDATE), ~.max(ARRIVALDATE)) > 365)	/Select records with more than 365 days interval for the same order
5	=A4.new(ID, ~.sum(AMOUNT):Amount)	/Create a sequence table, and count the sales amount of each order

A5	ID	Amount
	10998	6800.0
	11013	4560.0
	11032	20615.0
	…	…

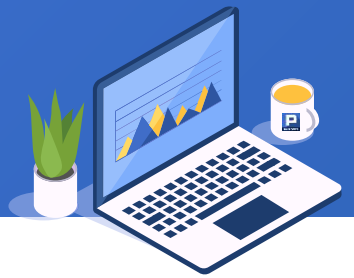
CONTENTS

1. NOT IN
2. NOT EXISTS
3. Double negative

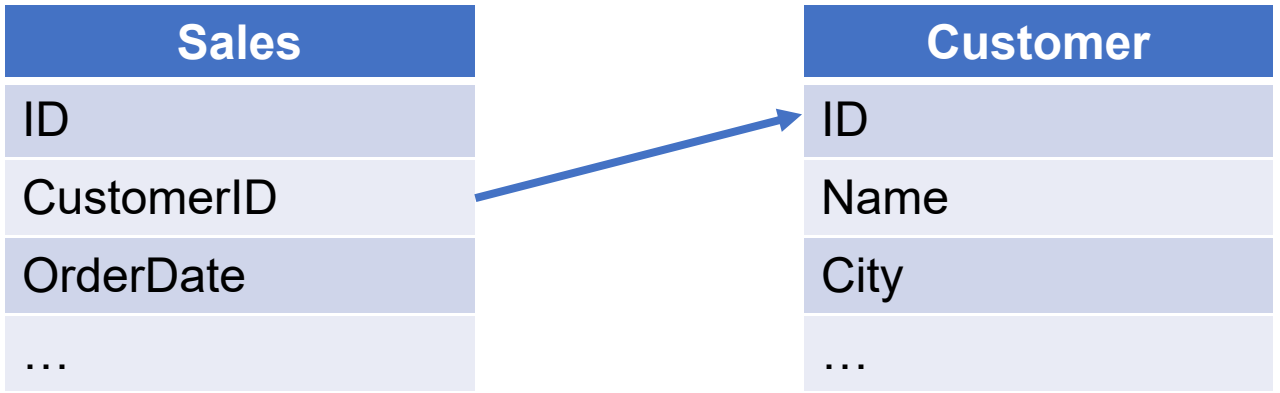
03

NOT IN , NOT EXISTS

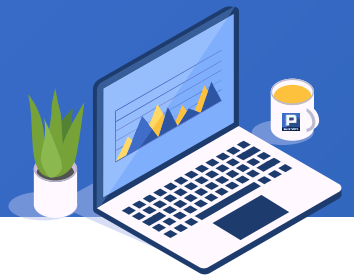
1. NOT IN



There are sales table and customer table. Query the sales of new customers in 2014.



1. NOT IN



The SQL statement is as follows:

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

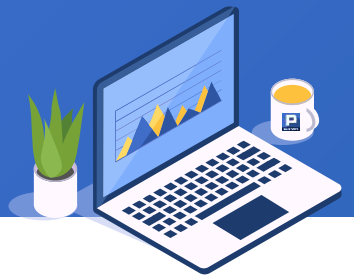
1. NOT IN

SPL is as follows, in which the @d option of A.join() function is used, and only unmatched records are retained:

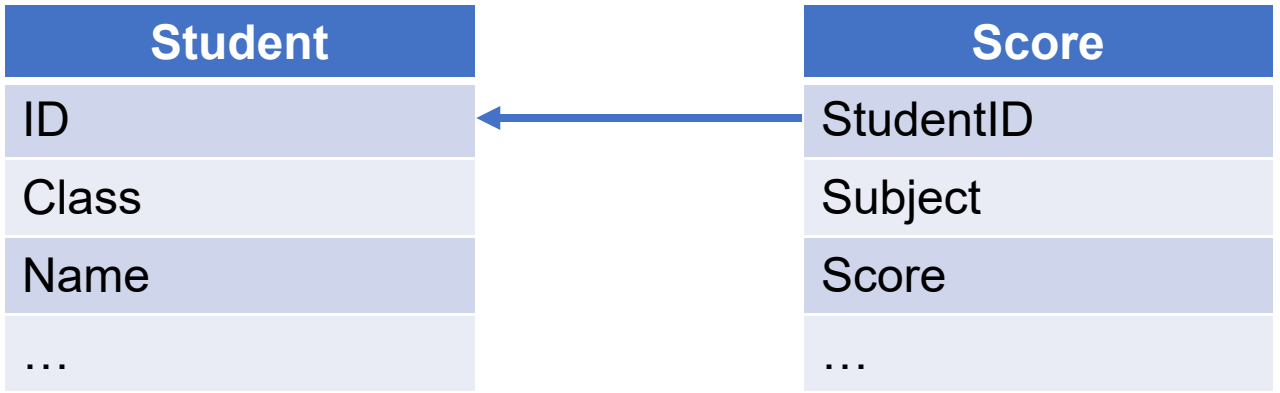
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Sales where year(OrderDate)=2014")	/Query 2014 sales records
3	=A1.query("select * from Customer")	/Query customer table
4	=A2.join@d(CustomerID ,A3:ID)	/Use A. join@d () to select the records whose customer ID does not exist in the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

A5	CustomerID	Amount
	DOS	11830.1
	HUN	57317.39
	…	…

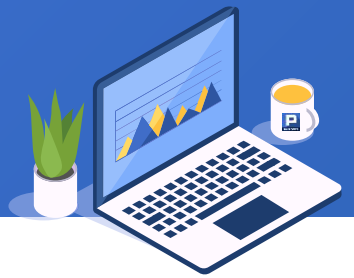
2. NOT EXISTS



Find students with scores above 80 in all subjects.



2. NOT EXISTS



The SQL statement is as follows:

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

✦ 2. NOT EXISTS

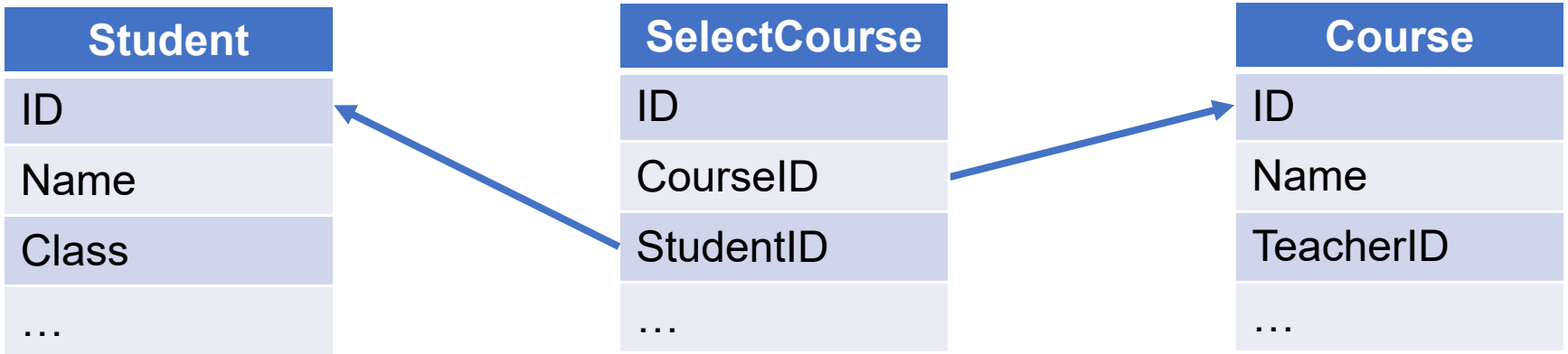
Only need to find students with no score lower or equal to 80. SPL is as follows:

	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Student")	/Query student table
3	=A1.query("select * from Score")	/Query score 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 unmatched records

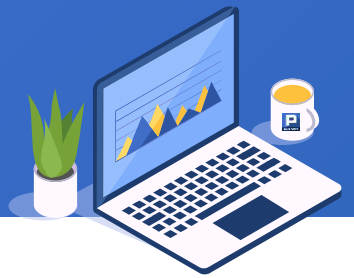
A6	ID	Class	Name
	2	Class 1	Ashley
	16	Class 2	Alexis

3. Double negative

Check the students who have taken all courses. The student table, course selection table and course table are as follows:



3. Double negative



Only need to find students with no any course that he has not selected. The SQL statement 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
        )
)
```

3. Double negative

SPL is as follows:

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

A7	ID	Name	Class
	4	Emily Smith	Class 1

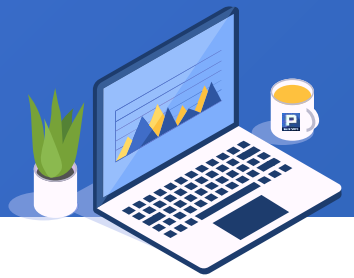
CONTENTS

1. ANY(or SOME)
2. ALL

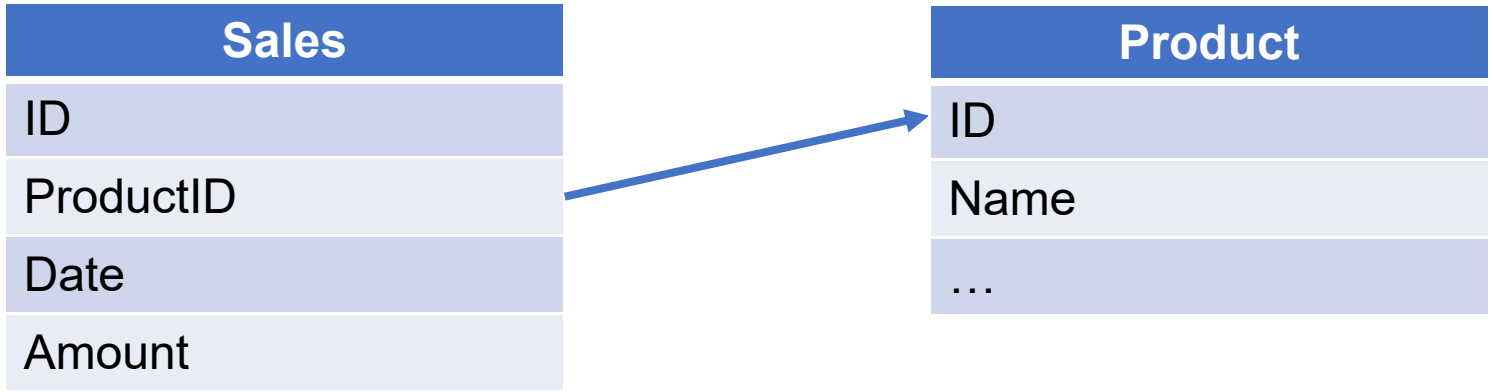
04

ANY , ALL

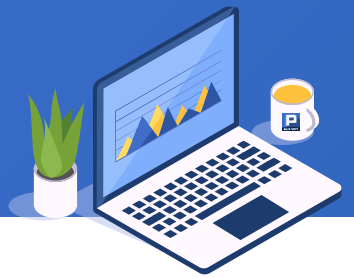
1. ANY(or SOME)



Query the total sales amount of products with the name including "water" sold each year. The sales table and product table are as follows:



1. ANY(or SOME)



The SQL statement is as follows:

```
select
        year(Date) as Year, sum(Amount) as Amount
from
        Sales
where
        ProductID = any ( select ID from Product where Name like '%water%' )
group by Year
```

When any is used with the equal sign, it has the same function as in.

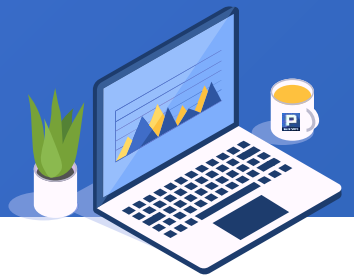
✦ 1. ANY(or SOME)

SPL is as follows:

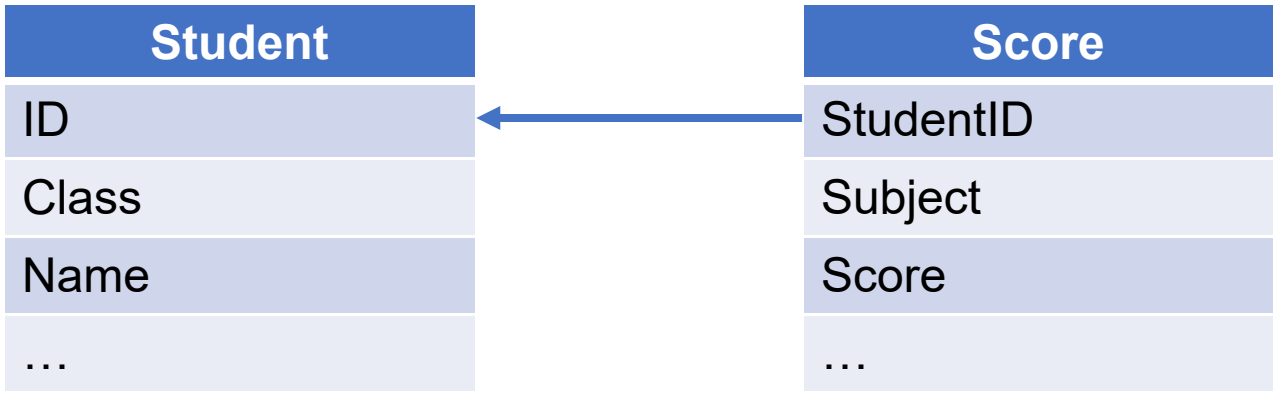
	A	B
1	=connect("db")	/Connect to database
2	=A1.query("select * from Sales")	/Query sales table
3	=A1.query("select * from Product")	/Query product table
4	=A3.select(like@c(Name,"*water*"))	/Find products with water in their product names
5	=A2.join@i(ProductID, A4:ID)	/Use A.join@i() function to join and filter
6	=A5.groups(year(Date):Year; sum(Amount):Amount)	/Group and summarize the total sales amount of each year

A6	Year	Amount
	2013	3271.0
	2014	6874.6
	…	…

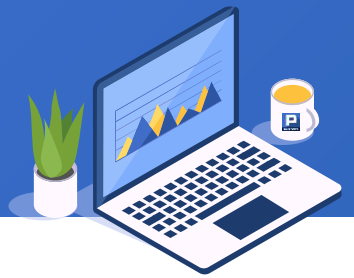
1. ANY(or SOME)



Look for students who are partial, that is, the score difference between the two subjects is more than 30 points. The student table and score table are as follows:



1. ANY(or SOME)



The SQL statement is as follows:

```
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
                        )
        )
```

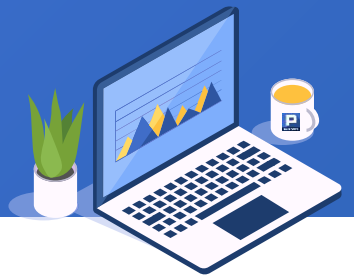
✦ 1. ANY(or SOME)

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

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

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

2. ALL



Take the employee table as an example to query which employees have higher salaries than all sales department employees.

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. ALL

The SQL statement is as follows:

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

2. ALL

When all is combined with a greater than sign, it is equivalent to max; when it is combined with a less than sign, it is equivalent to min. SPL is as follows:

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

THANKS