




SQL language: basics

The SELECT statement: basics




The SELECT statement: basics

- ⊃ Basic structure
- ⊃ The WHERE clause
- ⊃ Result ordering
- ⊃ Join
- ⊃ Aggregate functions
- ⊃ The GROUP BY operator


The SELECT statement: basics

Basic structure



The SELECT statement: example

- ⊃ Find the codes and the number of employees of the suppliers based in Paris



Supplier and part DB

P


PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

SP

SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

S

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens




The SELECT statement: example

- ⊃ Find the codes and the number of employees of the suppliers based in Paris

S

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

R

$$\pi_{SId, \#Employees} \left(\sigma_{City='Paris'} S \right)$$


The SELECT statement: example

Find the codes and the number of employees of the suppliers based in Paris

```
SELECT SId, #Employees
FROM S
WHERE City='Paris';
```

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SId	#Employees
S2	10
S3	30

DBG

Basic SELECT (no.1)

Find the codes of all products in the database

```
SELECT PId
FROM P;
```

PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

PId
P1
P2
P3
P4
P5
P6

DBG

Basic SELECT (no.2)

Find the codes of the products supplied by at least one supplier

```
SELECT PId
FROM SP;
```

SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

PId
P1
P2
P3
P4
P5
P6

DBG

Basic SELECT (no.2)

Find the codes of the products supplied by at least one supplier

```
SELECT PId
FROM SP;
```

It does not eliminate duplicates

DBG

Elimination of duplicates

DISTINCT keyword

- elimination of duplicates

Find the codes of the *distinct* products supplied by at least one supplier

DBG

Basic SELECT (no.2)

Find the codes of the *distinct* products supplied by at least one supplier

```
SELECT DISTINCT PId
FROM SP;
```

SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

PId
P1
P2
P3
P4
P5
P6

DBG

Selection of all information

Find all information related to products


```
SELECT PId, PName, Color, Weight, Store
FROM P;
```

or

```
SELECT *
FROM P;
```

R

PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London



Selection with an expression (1/3)

Find the codes of the products and the sizes expressed with the US standard

```
SELECT PId, Size-14
FROM P;
```


P

PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

➔

R


PId	Size
P1	26
P2	34
P3	34
P4	30
P5	26
P6	28



Selection with an expression (2/3)

Definition of a new *temporary* column for the computed expression

- the name of the temporary column may be defined by means of the AS keyword




Selection with an expression (3/3)

Find the codes of the products and the sizes expressed with the US standard

```
SELECT PId, Size-14 AS USSize
FROM P;
```


R

PId	USSize
P1	26
P2	34
P3	34
P4	30
P5	26
P6	28




Structure of the SELECT statement (1)

```
SELECT [DISTINCT] ListOfAttributesToDisplay
FROM ListOfTablesToUse;
```




The SELECT statement: basics

The WHERE clause



The WHERE clause

- ▷ It allows expressing selection conditions applied to each tuple individually
- ▷ A Boolean expression of predicates
- ▷ Simple predicates
 - comparison expressions between attributes and constants
 - text search
 - NULL values



The WHERE clause (no.1)


▷ Find the codes of the suppliers based in Paris

```
SELECT SId
FROM F
WHERE City='Paris';
```

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

→

SId
S2
S3



The WHERE clause (no.2)


▷ Find the codes and the number of employees of the suppliers that are not based in Paris

```
SELECT SId, #Employees
FROM S
WHERE City<>'Paris';
```

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

→

SId	#Employees
S1	20
S4	20
S5	30



Boolean expressions (no.1)


▷ Find the codes of the suppliers based in Paris that have more than 20 employees

```
SELECT SId
FROM S
WHERE City='Paris' AND #Employees>20;
```

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

→

SId
S3



Boolean expressions (no.2)


▷ Find the codes and the number of employees of the suppliers based in Paris or London

```
SELECT SId, #Employees
FROM S
WHERE City='Paris' OR City='London';
```

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

→

SId	#Employees
S1	2
S2	1
S3	3
S4	2



Boolean expressions (no.3)


▷ Find the codes and the number of employees of the suppliers based in Paris and in London

- the query may not be satisfied
- each supplier has only one city

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

→

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens




Text search

▷ LIKE operator

AttributeName LIKE CharacterString

- the `_` character represents a single arbitrary character (non-empty)
- the `%` character represents an arbitrary sequence of characters (possibly empty)



Text search (no.1)

▷ Find the codes and the names of the products whose name begins with the letter B


```
SELECT PId, PName
FROM P
WHERE PName LIKE 'B%';
```

P

PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	B blouse	Blue	48	Rome
P4	B blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

R


PId	PName
P3	B blouse
P4	B blouse



Text search (no.2)

▷ The Address attribute contains the string 'London'

Address LIKE '%London%'




Text search (no.3)

▷ The supplier identification number is 3 and

- it is preceded by a single unknown character
- it is exactly 2 characters long


SIId LIKE '_3'



Text search (no.4)

▷ The Store attribute does not have an 'e' in the second position

Store NOT LIKE '_e%'



Managing NULL values (no.1)

▷ Find the codes and the names of products with a size greater than 44


```
SELECT PId, PName
FROM P
WHERE Size > 44;
```

P

PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	NULL	Paris
P6	Shorts	Red	42	London


R

PId	PName
P2	Jeans
P3	Blouse



The NULL value


- ⊃ The tuples with a NULL size are not selected
 - the predicate `Size>44` evaluates to false
- ⊃ With NULL values, any comparison predicate is false



Searching for NULL values

- ⊃ IS special operator

AttributeName IS [NOT] NULL




Searching for NULL values (no.1)

- ⊃ Find the codes and the names of the products whose size is unknown

```
SELECT PId, PName
FROM P
WHERE Size IS NULL;
```

P				
PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	NULL	Paris
P6	Shorts	Red	42	London

R	
PId	PName
P5	Skirt




Searching for NULL values (no.2)

- ⊃ Find the codes and the names of products with a size greater than 44, or that may have a size greater than 44

```
SELECT PId, PName
FROM P
WHERE Size>44 OR Size IS NULL;
```


P				
PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	NULL	Paris
P6	Shorts	Red	42	London

R	
PId	PName
P2	Jeans
P3	Blouse
P5	Skirt




Structure of the SELECT statement (2)

```
SELECT [DISTINCT] ListOfAttributesToDisplay
FROM ListOfTablesToUse
[WHERE TupleConditions];
```



The SELECT statement: basics

Result ordering



Result ordering (no.1)

Find the codes of the products and their sizes, ordering the result by decreasing size

```
SELECT PId, Size
FROM P
ORDER BY Size DESC;
```

P				
PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

➔

R	
PId	Size
P2	48
P3	48
P4	44
P6	42
P1	40
P5	40

Ordering

ORDER BY clause

```
ORDER BY AttributeName [ASC | DESC]
        {, AttributeName [ASC | DESC]}
```

- the default ordering is ascending
 - if DESC is not specified
- the ordering attributes must appear in the SELECT clause
 - even implicitly (as in SELECT *)

Result ordering (no.2)

Find all information related to the products, ordering the result by increasing name order and decreasing size

```
SELECT PId, PName, Color, Size, Store
FROM P
ORDER BY PName, Size DESC;
```

P				
PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

Result ordering (no.2)

Find all information related to the products, ordering the result by increasing name order and decreasing size

```
SELECT PId, PName, Color, Size, Store
FROM P
ORDER BY PName, Size DESC;
```

R				
PId	PName	Color	Size	Store
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P2	Jeans	Green	48	Paris
P1	Jumper	Red	40	London
P6	Shorts	Red	42	London
P5	Skirt	Blue	40	Paris

Result ordering (no.2)

Find all information related to the products, ordering the result by increasing name order and decreasing size

```
SELECT *
FROM P
ORDER BY PName, Size DESC;
```

R				
PId	PName	Color	Size	Store
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P2	Jeans	Green	48	Paris
P1	Jumper	Red	40	London
P6	Shorts	Red	42	London
P5	Skirt	Blue	40	Paris

Result ordering (no.3)

Find the codes of the products and the sizes expressed with the US standard, ordering the result by increasing size

```
SELECT PId, Size-14 AS USSize
FROM P
ORDER BY USSize;
```


P				
PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

➔

R	
PId	USSize
P5	26
P1	26
P6	28
P4	30
P2	34
P3	34


Structure of the SELECT statement (3)

```
SELECT [DISTINCT] ListOfAttributesToDisplay
FROM ListOfTablesToUse
[WHERE TupleConditions ]
[ORDER BY ListOfOrderingAttributes];
```




The SELECT statement: basics

Join



Join (no.1)


⇒ Find the names of the suppliers that provide product P2



Supplier and part DB

S

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens




Supplier and part DB

S

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP


SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400



Cartesian product

⇒ Find the names of the suppliers that provide product P2

```
SELECT SName
FROM S, SP ;
```



Cartesian product

S.SId	S.SName	S.#Empl	S.City	SP.SId	SP.PId	SP.Qty
S1	Smith	20	London	S1	P1	300
S1	Smith	20	London	S1	P2	200
S1	Smith	20	London	S1	P3	400
S1	Smith	20	London	S1	P4	200
S1	Smith	20	London	S1	P5	100
S1	Smith	20	London	S1	P6	100
S1	Smith	20	London	S2	P1	300
...
S2	Jones	10	Paris	S1	P1	300
...
S2	Jones	10	Paris	S2	P1	300
...

Join (no.1)

=

S.SId	S.SName	S.#Empl	S.City	SP.SId	SP.PId	SP.Qty
S1	Smith	20	London	S1	P1	300
S1	Smith	20	London	S1	P2	200
S1	Smith	20	London	S1	P3	400
S1	Smith	20	London	S1	P4	200
S1	Smith	20	London	S1	P5	100
S1	Smith	20	London	S1	P6	100
S1	Smith	20	London	S2	P1	300
...
S2	Jones	10	Paris	S1	P1	300
...
S2	Jones	10	Paris	S2	P1	300
...

Join (no.1)

Find the names of the suppliers that provide product P2

```

SELECT SName
FROM S, SP
WHERE S.SId = SP.SId
    
```

↑
↑
 TableName.AttributeName

Join (no.1)

Find the names of the suppliers that provide product P2

```

SELECT SName
FROM S, SP
WHERE S.SId = SP.SId
    
```

Join condition

Join (no.1)

=

S.SId	S.SName	S.#Empl	S.City	SP.SId	SP.PId	SP.Qty
S1	Smith	20	London	S1	P1	300
S1	Smith	20	London	S1	P2	200
S1	Smith	20	London	S1	P3	400
S1	Smith	20	London	S1	P4	200
S1	Smith	20	London	S1	P5	100
S1	Smith	20	London	S1	P6	100
S1	Smith	20	London	S2	P1	300
...
S2	Jones	10	Paris	S1	P1	300
...
S2	Jones	10	Paris	S2	P1	300
...


Join (no.1)

S.SId	S.SName	S.#Empl	S.City	SP.SId	SP.PId	SP.Qty
S1	Smith	20	London	S1	P1	300
S1	Smith	20	London	S1	P2	200
S1	Smith	20	London	S1	P3	400
S1	Smith	20	London	S1	P4	200
S1	Smith	20	London	S1	P5	100
S1	Smith	20	London	S1	P6	100
S2	Jones	10	Paris	S2	P1	300
S2	Jones	10	Paris	S2	P2	400
S3	Blake	30	Paris	S3	P2	200
S4	Clark	20	London	S4	P3	200
S4	Clark	20	London	S4	P4	300
S4	Clark	20	London	S4	P5	400

Join (no.1)

Find the names of the suppliers that provide product P2

```
SELECT SName
FROM S, SP
WHERE S.SId=SP.SId AND
      PId='P2';
```



Join (no.1)

SP.PId='P2'

S.SId	S.SName	S.#Empl	S.City	SP.SId	SP.PId	SP.Qty
S1	Smith	20	London	S1	P1	300
S1	Smith	20	London	S1	P2	200
S1	Smith	20	London	S1	P3	400
S1	Smith	20	London	S1	P4	200
S1	Smith	20	London	S1	P5	100
S1	Smith	20	London	S1	P6	100
S2	Jones	10	Paris	S2	P1	300
S2	Jones	10	Paris	S2	P2	400
S3	Blake	30	Paris	S3	P2	200
S4	Clark	20	London	S4	P3	200
S4	Clark	20	London	S4	P4	300
S4	Clark	20	London	S4	P5	400



Join (no.1)

S.SId	S.SName	S.#Empl	S.City	SP.SId	SP.PId	SP.Qty
S1	Smith	20	London	S1	P2	200
S2	Jones	10	Paris	S2	P2	400
S3	Blake	30	Paris	S3	P2	200




Join (no.1)

Find the names of the suppliers that provide product P2

R

SName
Smith
Jones
Blake



Join (no.1)


Find the names of the suppliers that provide product P2

- in relational algebra

$$\pi_{SName} \left(\sigma_{PId='P2'} \left(S \bowtie SP \right) \right)$$

↔

$$\pi_{SName} \left(S \bowtie \sigma_{PId='P2'}(SP) \right)$$



Join (no.1)


Find the names of the suppliers that provide product P2

```
SELECT SName
FROM S, SP
WHERE S.SId=SP.SId
      AND PId='P2';
```

↔

```
SELECT SName
FROM S, SP
WHERE PId='P2' AND
      S.SId=SP.SId;
```

The result and the efficiency are independent of the order of predicates in the WHERE clause



Join (no.1)


Find the names of the suppliers that provide product P2

```
SELECT SName
FROM S, SP
WHERE S.SId=SP.SId
AND PId='P2';
```

↔

```
SELECT SName
FROM SP, S
WHERE S.SId=SP.SId
AND PId='P2';
```


The result and the efficiency are independent of the order of tables in the FROM clause



Join (no.1)

Declarativity of the SQL language

- in relational algebra we define the order in which operators are applied
- in SQL the best order is chosen by the optimizer independently of
 - the order of conditions in the WHERE clause
 - the order of tables in the FROM clause




Join (no.2)

Find the names of the suppliers that supply at least one red product

```
SELECT SName
FROM S, SP, P
WHERE S.SId=SP.SId AND P.PId=SP.PId
AND Color='Red';
```

FROM clause with N tables

- at least N-1 join conditions in the WHERE clause




Join (no.3)

Find the pairs of supplier codes such that both suppliers are based in the same city

```
SELECT SX.SId, SY.SId
FROM S AS SX, S AS SY
WHERE SX.City=SY.City;
```

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens



Join (no.3)


Find the pairs of supplier codes such that both suppliers are based in the same city

```
SELECT SX.SId, SY.SId
FROM S AS SX, S AS SY
WHERE SX.City=SY.City;
```

The result includes

- pairs of identical values
- permutations of the same pairs of values

SX.SId	SY.SId
S1	S1
S1	S4
S2	S2
S2	S3
S3	S2
S3	S3
S4	S1
S4	S4
S5	S5




Join (no.3)

Find the pairs of supplier codes such that both suppliers are based in the same city

```
SELECT SX.SId, SY.SId
FROM S AS SX, S AS SY
WHERE SX.City=SY.City AND
SX.SId <> SY.SId;
```

It removes pairs of identical values

SX.SId	SY.SId
S1	S4
S2	S2
S2	S3
S3	S2
S3	S3
S4	S1
S4	S4
S5	S5




Join (no.3)

Find the pairs of supplier codes such that both suppliers are based in the same city

```
SELECT SX.SId, SY.SId
FROM S AS SX, S AS SY
WHERE SX.City=SY.City AND
      SX.SId < SY.SId;
```

It eliminates the permutations of the same pairs of values

FX.CodF	FY.CodF
F1	F1
F1	F4
F2	F2
F2	F3
F3	F2
F3	F3
F4	F1
F4	F4
F5	F5




Join (no.3)

Find the pairs of supplier codes such that both suppliers are based in the same city


```
SELECT SX.SId, SY.SId
FROM S AS SX, S AS SY
WHERE SX.City=SY.City AND
      SX.SId < SY.SId;
```

SX.SId	SY.SId
S1	S4
S2	S3



Join: alternative syntax


- ▷ Different types of join may be specified
 - outer join
- ▷ It allows differentiating between
 - join conditions and
 - tuple selection conditions
- ▷ Introduced in SQL-2
 - not widely available in commercial products



Join: alternative syntax

```
SELECT [DISTINCT] Attributes
FROM Table JoinType JOIN Table ON
      JoinCondition
[WHERE TupleConditions];
```


▷ *JoinType* = < INNER | [FULL | LEFT | RIGHT] OUTER >



INNER join

Find the names of the suppliers that supply at least one red product


```
SELECT SName
FROM P INNER JOIN SP ON P.PId=SP.PId
      INNER JOIN S ON S.SId=SP.SId
WHERE P.Color='Red';
```



OUTER join

Find the codes and the names of the suppliers together with the codes of the products they provide, also including the suppliers that are not supplying any product

```
SELECT S.SId, SName, PId
FROM S LEFT OUTER JOIN SP ON
      S.SId=SP.SId;
```



OUTER join

R

S.SId	S.SName	SP.PId
S1	Smith	P1
S1	Smith	P2
S1	Smith	P3
S1	Smith	P4
S1	Smith	P5
S1	Smith	P6
S2	Jones	P1
S2	Jones	P2
S3	Blake	P2
S4	Clark	P3
S4	Clark	P4
S4	Clark	P5
S5	Adams	NULL

DBG

The SELECT statement: basics

Aggregate functions

DBG

Aggregate functions

⇒ An aggregate function

- operates on a set of values
- produces a single (aggregate) value as a result

DBG

Aggregate functions

⇒ Aggregate functions available in SQL-2

- COUNT: count of elements in a given attribute
- SUM: sum of values for a given attribute
- AVG: average of values for a given attribute
- MAX: maximum value of a given attribute
- MIN: minimum value of a given attribute

DBG

Aggregate functions

⇒ An aggregate function

- operates on a set of values
- produces a single (aggregate) value as a result
- is specified in the SELECT clause

DBG

Structure of the SELECT statement (4)


```
SELECT ListOfAggregateFunctionsToDisplay
FROM ListOfTablesToUse
[WHERE TupleConditions ]
[ORDER BY ListOfOrderingAttributes];
```

DBG

Aggregate functions

▷ An aggregate function

- operates on a set of values
- produces a single (aggregate) value as a result
- is specified in the SELECT clause
 - non-aggregate attributes may not be specified at the same time
 - multiple aggregate functions may be specified simultaneously




The COUNT function

▷ Counts the number of elements in a set

- rows in a table
- (possibly distinct) values for one or more attributes

COUNT (< * | [DISTINCT | ALL] ListOfAttributes >)



The COUNT function (no.1)


▷ Find the number of suppliers

SELECT COUNT(*)
FROM S;

S			
SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

→

R
5



The COUNT function (no.2)

▷ Find the number of suppliers that supply at least one product


SP		
SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

SELECT COUNT(*)
FROM SP;

→

R
12

▷ It counts the number of supplied products, not the suppliers



The COUNT function (no.2)

▷ Find the number of suppliers that supply at least one product


SP		
SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

SELECT COUNT(SId)
FROM SP;

→

R
12

▷ It still counts the number of supplied products, not the suppliers



The COUNT function (no.2)

▷ Find the number of suppliers that supply at least one product


SP		
SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

SELECT COUNT(DISTINCT SId)
FROM SP;

→

R
4

▷ It counts the number of distinct suppliers




The COUNT function

- Counts the number of elements in a set
 - rows in a table
 - (possibly distinct) values for one or more attributes

COUNT (< * | [DISTINCT | ALL] ListOfAttributes >)

- If the function argument is preceded by **DISTINCT**, it counts the number of distinct values of the argument



Aggregate functions and WHERE

- Find the number of suppliers providing product P2

SP	SId	PId	Qty
	S1	P1	300
	S1	P2	200
	S1	P3	400
	S1	P4	200
	S1	P5	100
	S1	P6	100
	S2	P1	300
	S2	P2	400
	S3	P2	200
	S4	P3	200
	S4	P4	300
	S4	P5	400


```

SELECT COUNT(*)
FROM SP
WHERE PId='P2';
    
```

SId	PId	Qty
S1	P2	200
S2	P2	400
S3	P2	200


➔

R
3




Aggregate functions and WHERE

- Aggregate functions are only evaluated once all predicates in the **WHERE** clause have been applied



The SUM, MAX, MIN, AVG functions

- **SUM, MAX, MIN and AVG**
 - they allow an attribute or an expression as argument
- **SUM and AVG**
 - they only allow numeric type or time interval attributes
- **MAX and MIN**
 - they require an expression that can be ordered
 - may also be applied to character strings and time instants



The SUM function

- Find the overall quantity of supplied pieces for product P2

SP	SId	PId	Qty
	S1	P1	300
	S1	P2	200
	S1	P3	400
	S1	P4	200
	S1	P5	100
	S1	P6	100
	S2	P1	300
	S2	P2	400
	S3	P2	200
	S4	P3	200
	S4	P4	300
	S4	P5	400


```

SELECT SUM(Qty)
FROM SP
WHERE PId='P2';
    
```

SId	PId	Qty
S1	P2	200
S2	P2	400
S3	P2	200


➔

R
800



The SELECT statement: basics

The GROUP BY operator



Grouping

➤ For each product, find the overall quantity of supplied pieces

SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

➔

SId	PId	Qty
S1	P1	300
S2	P1	300
S1	P2	200
S2	P2	400
S3	P2	200
S1	P3	400
S4	P3	200
S1	P4	200
S4	P4	300
S1	P5	100
S4	P5	400
S1	P6	100

➔

PId	Qty
P1	600
P2	800
P3	600
P4	500
P5	500
P6	100

Grouping

➤ For each product, find the overall quantity of supplied pieces

```
SELECT PId, SUM(Qty)
FROM SP
GROUP BY PId;
```

GROUP BY

➤ Grouping clause

GROUP BY *ListOfGroupingAttributes*

- the order of grouping attributes is irrelevant

➤ Only

- attributes specified in the GROUP BY clause
- aggregate functions

are allowed to appear in the SELECT statement

GROUP BY and WHERE

➤ For each product, find the overall quantity of pieces supplied by suppliers based in Paris

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

GROUP BY and WHERE

➤ For each product, find the overall quantity of pieces supplied by suppliers based in Paris

```
SELECT ...
FROM SP, S
WHERE SP.SId=S.SId AND City='Paris'
...
```

GROUP BY and WHERE

➤ For each product, find the overall quantity of pieces supplied by suppliers based in Paris

S.SId	S.SName	S.#Empl	S.City	SP.SId	SP.PId	SP.Qty
S1	Smith	20	London	S1	P1	300
S1	Smith	20	London	S1	P2	200
S1	Smith	20	London	S1	P3	400
S1	Smith	20	London	S1	P4	200
S1	Smith	20	London	S1	P5	100
S1	Smith	20	London	S1	P6	100
S2	Jones	10	Paris	S2	P1	300
S2	Jones	10	Paris	S2	P2	400
S3	Blake	30	Paris	S3	P2	200
S4	Clark	20	London	S4	P3	200
S4	Clark	20	London	S4	P4	300
S4	Clark	20	London	S4	P5	400

GROUP BY and WHERE

▷ For each product, find the overall quantity of pieces supplied by suppliers based in Paris

```

SELECT PId, SUM(Qty)
FROM SP, S
WHERE SP.SId=S.SId AND City='Paris'
GROUP BY PId;
    
```

▷ Products that are not supplied by any supplier are not included in the result




GROUP BY and WHERE

▷ For each product, find the overall quantity of pieces supplied by suppliers based in Paris

SP.PId	SP.Qty
P1	300
P2	400
P2	200

➔

R	
SP.PId	
P1	300
P2	600



GROUP BY and SELECT


▷ For each product, find the code, the name and the overall supplied quantity

```

SELECT P.PId, PName, SUM(Qty)
FROM P, SP
WHERE P.PId=SP.PId
GROUP BY P.PId, PName
    
```

▷ Syntactic device


- attributes that are unambiguously determined by other attributes already present in the GROUP BY clause may be added *without altering the result*



Structure of the SELECT statement (5)

```

SELECT [DISTINCT] ListOfAttributesToDisplay
FROM ListOfTablesToUse
[WHERE TupleConditions ]
[GROUP BY ListOfGroupingAttributes ]
[ORDER BY ListOfOrderingAttributes ];
    
```




Group selection condition

▷ Find the overall quantity of supplied pieces for the products for which at least 600 pieces are supplied *overall*

- the condition is defined on *aggregate values*

▷ The WHERE clause may not be used for this purpose



Group selection condition (no.1)

▷ Find the overall quantity of supplied pieces for the products for which at least 600 pieces are supplied *overall*


SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400
S4	P5	400
S4	P5	400

➔

SId	PId	Qty
S1	P1	300
S2	P1	300
S1	P2	200
S2	P2	400
S3	P2	200
S1	P3	400
S4	P3	200
S1	P4	200
S4	P4	300
S1	P5	100
S4	P5	400
S1	P6	100

➔

R	
PId	
P1	600
P2	800
P3	600




Group selection condition (no.1)

⇒ Find the overall quantity of supplied pieces for the products for which at least 600 pieces are supplied *overall*

```
SELECT PId, SUM(Qty)
FROM SP
GROUP BY PId
HAVING SUM(Qty)>=600;
```

⇒ The HAVING clause allows the specification of conditions on the aggregate functions




Group selection condition (no.2)

⇒ Find the codes of the red products supplied by more than one supplier

P				
PId	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London


SP		
SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400



Group selection condition (no.2)

⇒ Find the codes of the red products supplied by more than one supplier

```
SELECT SP.PId
FROM SP, P
WHERE SP.PId=P.PId AND Color='Red'
GROUP BY SP.PId
HAVING COUNT(*)>1;
```




Group selection condition (no.2)

⇒ Find the codes of the red products supplied by more than one supplier

S.SId	S.PId	S.Qty	P.PId	P.PName	P.Color	P.Size	P.Store
S1	P1	300	P1	Jumper	Red	40	London
S2	P1	300	P1	Jumper	Red	40	London
S1	P6	100	P6	Shorts	Red	42	London

↓

R
PId
P1



Structure of the SELECT statement

```
SELECT [DISTINCT] ListOfAttributesToDisplay
FROM ListOfTablesToUse
[WHERE TupleConditions ]
[GROUP BY ListOfGroupingAttributes ]
[HAVING AggregateConditions ]
[ORDER BY ListOfOrderingAttributes ];
```

