



Politecnico  
di Torino



# SQL language: basics

---

# SQL language: basics

---

- SQL Language
- Language Instruction
- Sample notation and database
- SELECT Statement
- Aggregate Functions
- Operator GROUP BY

# The SQL language

---

- A language for managing relational databases
  - Structured Query Language
- SQL provides commands to
  - define the schema of a relational database
  - read and write data
  - define the schema of derived tables
  - define user access privileges
  - manage transactions
- The SQL language may be used in two ways
  - interactive
  - compiled
    - a host language encapsulates the SQL commands
    - SQL commands can be distinguished from the host language commands by means of appropriate syntactic mechanisms

# The SQL language

---

- SQL is a *set-level* language
  - operators are applied to relations (tables)
  - the result is always a relation (table)
- SQL is a *declarative* language
  - it describes *what to do* and not how to do it
  - it has a higher level of abstraction compared to traditional programming languages

# SQL instructions

---

The SQL language

# The SQL language

---

- Can be divided into
  - **DML** (Data Manipulation Language)
    - language for querying and updating the data
  - **DDL** (Data Definition Language)
    - language for defining the database structure

# Data Manipulation Language

---

- To query a database in order to extract data of interest
  - SELECT
- To modify a database instance
  - INSERT: insertion of new information into a table
  - UPDATE: update of the information in the database
  - DELETE: cancellazione di dati obsoleti

# Data Definition Language

---

- To define a database schema
  - creation, modification and deletion of tables: CREATE, ALTER, DROP TABLE
- To define derived tables
  - creation, modification and deletion of tables whose content is obtained from other database tables: CREATE, ALTER, DROP VIEW
- To define complementary data structures for efficiently retrieving the data
  - creation and deletion of indices: CREATE, DROP INDEX
- To define user access privileges
  - grant and revocation of privileges on resources: GRANT, REVOKE
- To define transactions
  - termination of a transaction: COMMIT, ROLLBACK



# Notation and example database

---

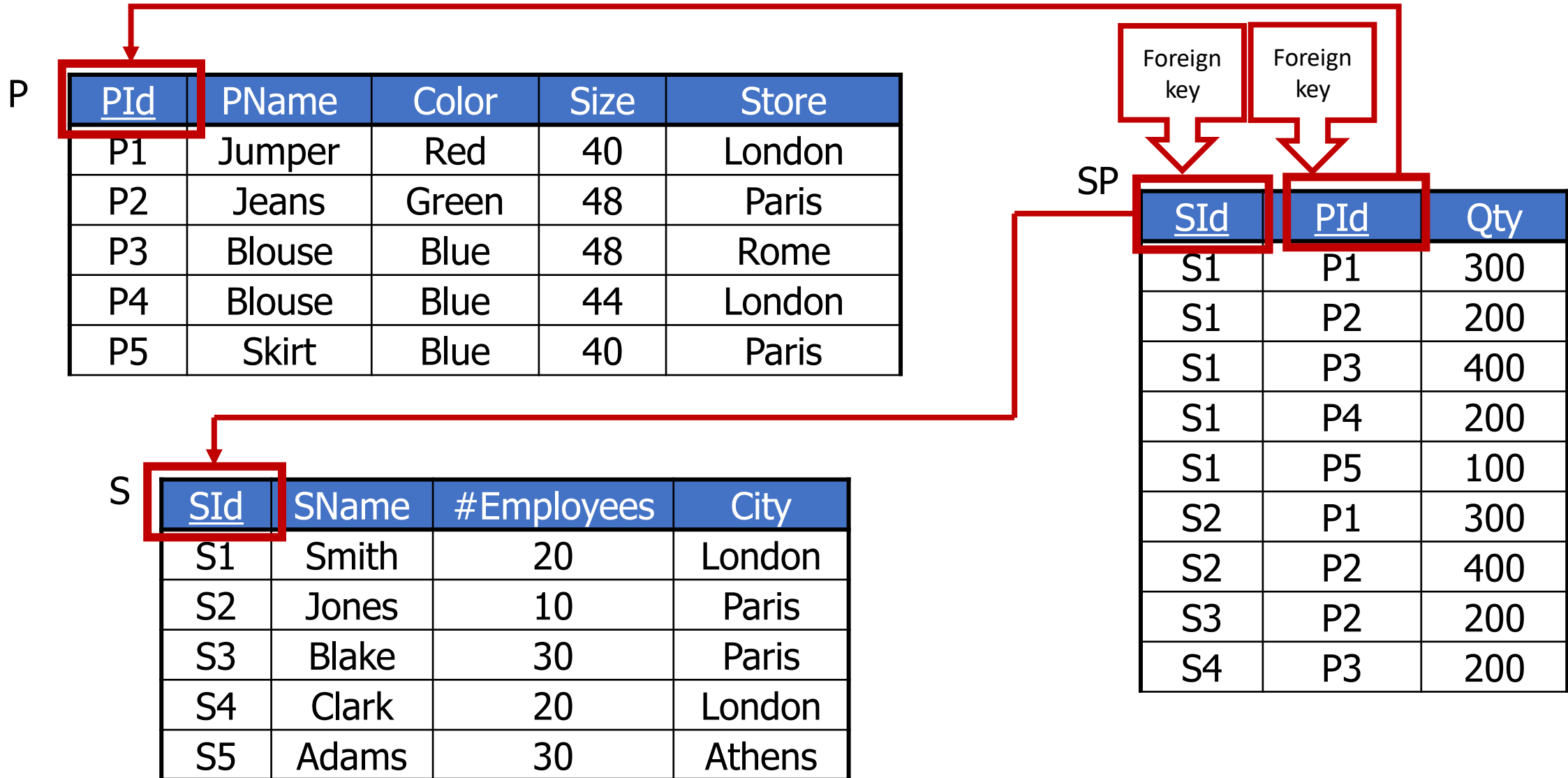
The SQL Language

# Syntax of SQL commands

---

- Notation
  - language keywords
    - upper case
  - variable terms
- Grammar
  - angle brackets `< >`
    - to isolate a syntactic term
  - square brackets `[ ]`
    - the enclosed term is optional
  - braces `{ }`
    - the enclosed term may not appear or may be repeated an arbitrary number of items
  - vertical bar `|`
    - a term must be chosen among the options separated by the vertical bars

# Example database: Supply-Product



# Example database: Supply-Product

---

- Supplier and part DB
  - table P describes the available products
    - primary key: PId
  - table S describes the suppliers
    - primary key: SId
  - table SP describes supplies, by relating each product to the suppliers that provide it
    - primary key: (SId, PId)
    - PId: Foreign key (SP) REFERENCES PId(P)
    - SId: Foreign key (SP) REFERENCES SId(S)

# The SELECT statement: basics

---

The SQL language

# SELECT

---

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

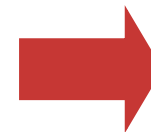
# Basic SELECT(n.1)

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

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

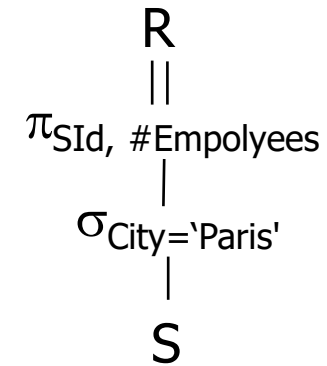
S

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



R

SId	#Employees
S2	10
S3	30



# Basic SELECT(n.2)

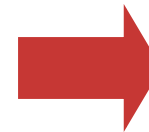
- Find the codes of all products in the database

```
SELECT PId
FROM P;
```

$$\begin{array}{c} R \\ || \\ \pi_{PId} \\ | \\ P \end{array}$$

P

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



R

<u>PId</u>
P1
P2
P3
P4
P5
P6



# Basic SELECT(n.3)

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

SP

<u>SId</u>	<u>PId</u>	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 PId  
FROM SP;



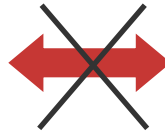
R

<u>PId</u>
P1
P2
P3
P4
P5
P6
P1
P2
P2
P3
P4
P5

# Basic SELECT(n.3)

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

SELECT PId  
FROM SP;



R  
||  
 $\pi_{PId}$   
|  
SP

- It does not eliminate duplicates

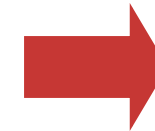
# Elimination of duplicates: DISTINCT

- **DISTINCT** keyword allows the elimination of duplicates
- Find the codes of the *distinct* products supplied by at least one supplier

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

SP

<u>SId</u>	<u>PID</u>	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



R

PId
P1
P2
P3
P4
P5
P6

# Selection of all information

- Find **all** information related to products

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

or

```
SELECT *  
FROM P;
```

R

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

# Selection with an expression

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

```
SELECT PId, Size-14 [AS USSize]  
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	Blue	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London



R

PId	USSize
P1	26
P2	34
P3	34
P4	30
P5	26
P6	38

- 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

# The WHERE clause

- It allows expressing selection conditions applied to each tuple individually
- A Boolean expression composed by one or more predicates
- Simple predicates
  - comparison between attributes and constants
  - text search
  - NULL values

# The WHERE clause (n.1)

- Find the codes of the suppliers based in Paris

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

F

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



R

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';
```

F

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

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;
```

S

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



R

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';
```

S

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



R

SId	#Employees
F1	20
F2	10
F3	30
F4	20

# 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

S

<u>SId</u>	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	Blouse	Blue	48	Rome
P4	Blouse	Blue	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London



R

PId	PName
P3	Blouse
P4	Blouse

# Text search (no.2)

---

- The Address attribute contains the string 'London'

Address LIKE '%London%'

- The supplier identification number is 3 and
  - it is preceded by a single unknown character
  - it is exactly 2 characters long

SIId LIKE ` \_3'

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

Store NOT LIKE '\_e%'

# Searching for NULL values

- **IS** special operator

*AttributeName* **IS [NOT] NULL**

- With **NULL** values, any comparison predicate is false


# Managing NULL values

- 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	Blue	44	London
P5	Skirt	Blue	NULL	Paris
P6	Shorts	Red	42	London



R

PId	PName
P2	Jeans
P3	Blouse

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



# 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	Blue	44	London
P5	Skirt	Blue	NULL	Paris
P6	Shorts	Red	42	London



R

PId	PName
P5	Skirt

# Searching for NULL values(n.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	Blue	44	London
P5	Skirt	Blue	NULL	Paris
P6	Shorts	Red	42	London



R

PId	Pname
P2	Jeans
P3	Blouse
P5	Skirt

# Result 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.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	Blue	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

# Result ordering (no.2)

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

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

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

R

<u>PId</u>	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	Blue	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London



R

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

# Join

- Defined by the **FROM** and **WHERE** clauses
- The result and efficiency of the query
  - are independent of the order of the tables in the FROM clause
  - are independent of the predicate order in the WHERE clause
  - the optimal execution order is selected by the DBMS (optimizer module)
- FROM clause with **N** Tables
  - at least **N-1** join conditions in the WHERE clause

# Join (n.1)

- Find the names of the suppliers that provide product P2

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

<u>SId</u>	<u>PID</u>	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



# Cartesian product

---

- Find the names of the suppliers that provide product P2

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

# Cartesian product

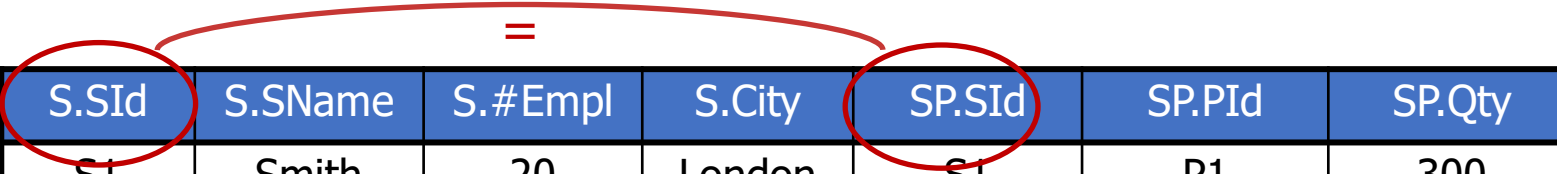
S.SId	S.SName	S.#Empl	S.City	SP.SId	SP.PId	SP.Qty
S1	Smith	20	London	S1	P1	300
S1	Smith	20	London	S1	P2	200
S1	Smith	20	London	S1	P3	400
S1	Smith	20	London	S1	P4	200
S1	Smith	20	London	S1	P5	100
S1	Smith	20	London	S1	P6	100
S1	Smith	20	London	S2	P1	300
...	...	...	...	...	...	...
S2	Jones	10	Paris	S1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

# Join (n.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 (n.1)



A red circle highlights the 'S.SId' column header and the 'SP.SId' column header. A red line with an equals sign (=) connects the two circles, indicating a join condition between these two columns.

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 (n.1)

- Find the names of the suppliers that provide product P2

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

Join condition

TableName.AttributeName

# Join (n.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 condition

↑ ↑  
TableName.AttributeName

# Join (n.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 (n.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



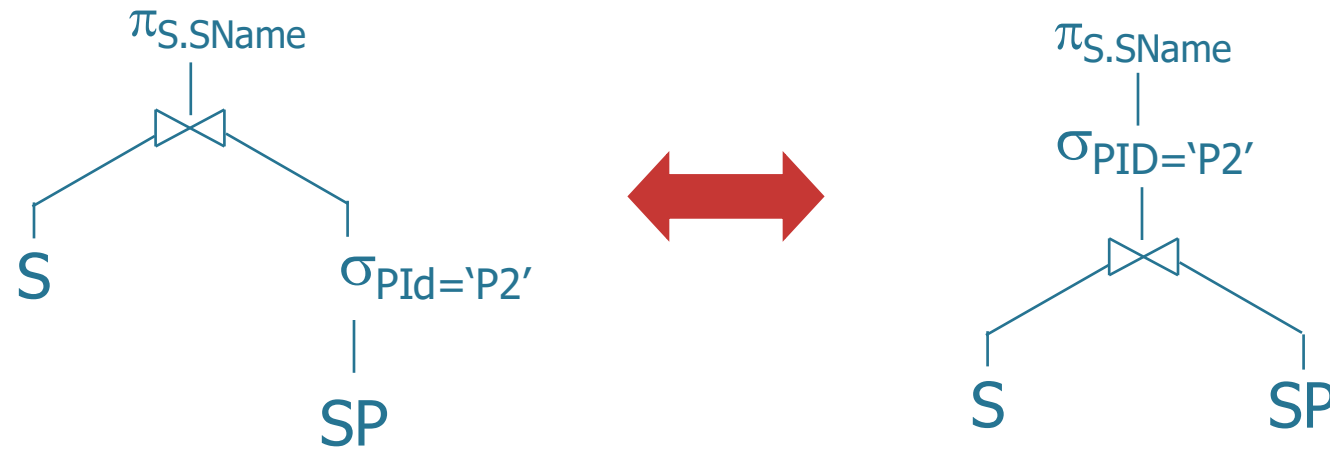
R

SName
Smith
Jones
Blake



# Join (n.1)

- Find the names of the suppliers that provide product P2
  - in relational algebra



# Join (n.1)

---

- Find the names of the suppliers that provide product P2
  - in relational algebra

```
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 efficiency are independent
  - from the order of the predicates in the WHERE clause
  - from the order of the tables in the FROM clause

# SQL Declarability

---

- In relational algebra (procedural language) we define the order in which the operators are applied
- In SQL (declarative language) the best order is chosen by the optimizer independently
  - from the order of the conditions in the WHERE clause
  - from the order of the tables in the FROM clause

# Join (n.2)

---

- Find the name of suppliers who provide 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 (n.2)

- 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;
```

S AS SX

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

S AS SY

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

# Join (n.2)

- 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

R

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

# Join (n.2)

- 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

R

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

# Join (n.2)

- 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

R

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

R

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

```
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;
```

S.Sid	S.SName	SP.Sid
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
<i>S5</i>	<i>Adams</i>	<i>NULL</i>

# Aggregate Functions

---

Introduction to SQL

# Aggregate function

---

- It operates on a set of values
- It produces a single (aggregate) value as a result
- It is specified in the **SELECT** clause
  - non-aggregate attributes may not be specified at the same time
  - multiple aggregate functions may be specified simultaneously
- Aggregate functions are only evaluated once all predicates in the **WHERE** clause have been applied

# Aggregate functions

---

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

# COUNT

- 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

# The COUNT function (n.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



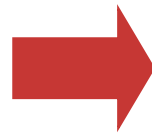
# The COUNT function (n.2)

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

SP

<u>SId</u>	<u>PId</u>	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

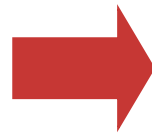
# The COUNT function (n.2)

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

SP

<u>SId</u>	<u>PId</u>	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

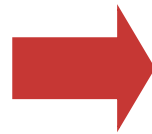
# The COUNT function (n.2)

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

SP

<u>SId</u>	<u>PId</u>	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

# Aggregate functions and WHERE

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

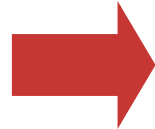
# Aggregate functions and WHERE

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

# SUM, MAX, MIN, AVG

- **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 GROUP BY operator

---

Introduction to SQL

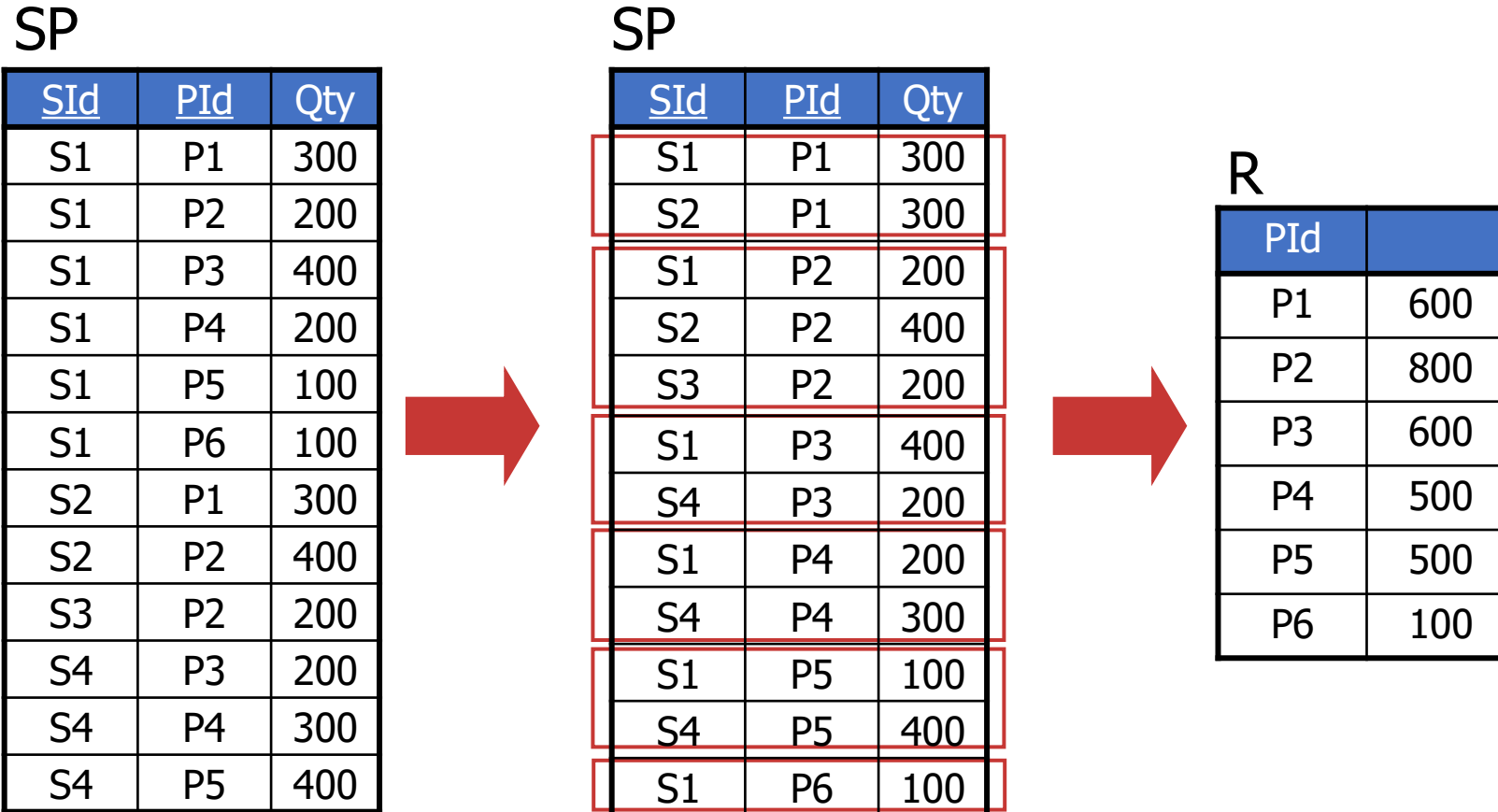


# GROUP BY

- Grouping clause  
**GROUP BY** ListOfGroupingAttributes
  - The order of grouping attributes is irrelevant
- In the SELECT statement **only**
  - attributes specified in the GROUP BY clause
  - aggregate functionsare allowed to appear
- Attributes that are unambiguously determined by other attributes already present in the GROUP BY clause may be added *without altering the result*

# Grouping

- *For each product*, find the overall quantity of supplied pieces



# Grouping

- *For each product*, find the overall quantity of supplied pieces

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



SP

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
P4	500
P5	500
P6	100

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

# GROUP BY and WHERE

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

S

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

SP

<u>SId</u>	<u>PId</u>	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.#Employees	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
<i>S2</i>	<i>Jones</i>	<i>10</i>	<i>Paris</i>	<i>S2</i>	<i>P1</i>	<i>300</i>
<i>S2</i>	<i>Jones</i>	<i>10</i>	<i>Paris</i>	<i>S2</i>	<i>P2</i>	<i>400</i>
<i>S3</i>	<i>Blake</i>	<i>30</i>	<i>Paris</i>	<i>S3</i>	<i>P2</i>	<i>200</i>
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, F
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 S.PId=SP.PId
GROUP BY P.PId, PName
```

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

# Group selection condition: HAVING

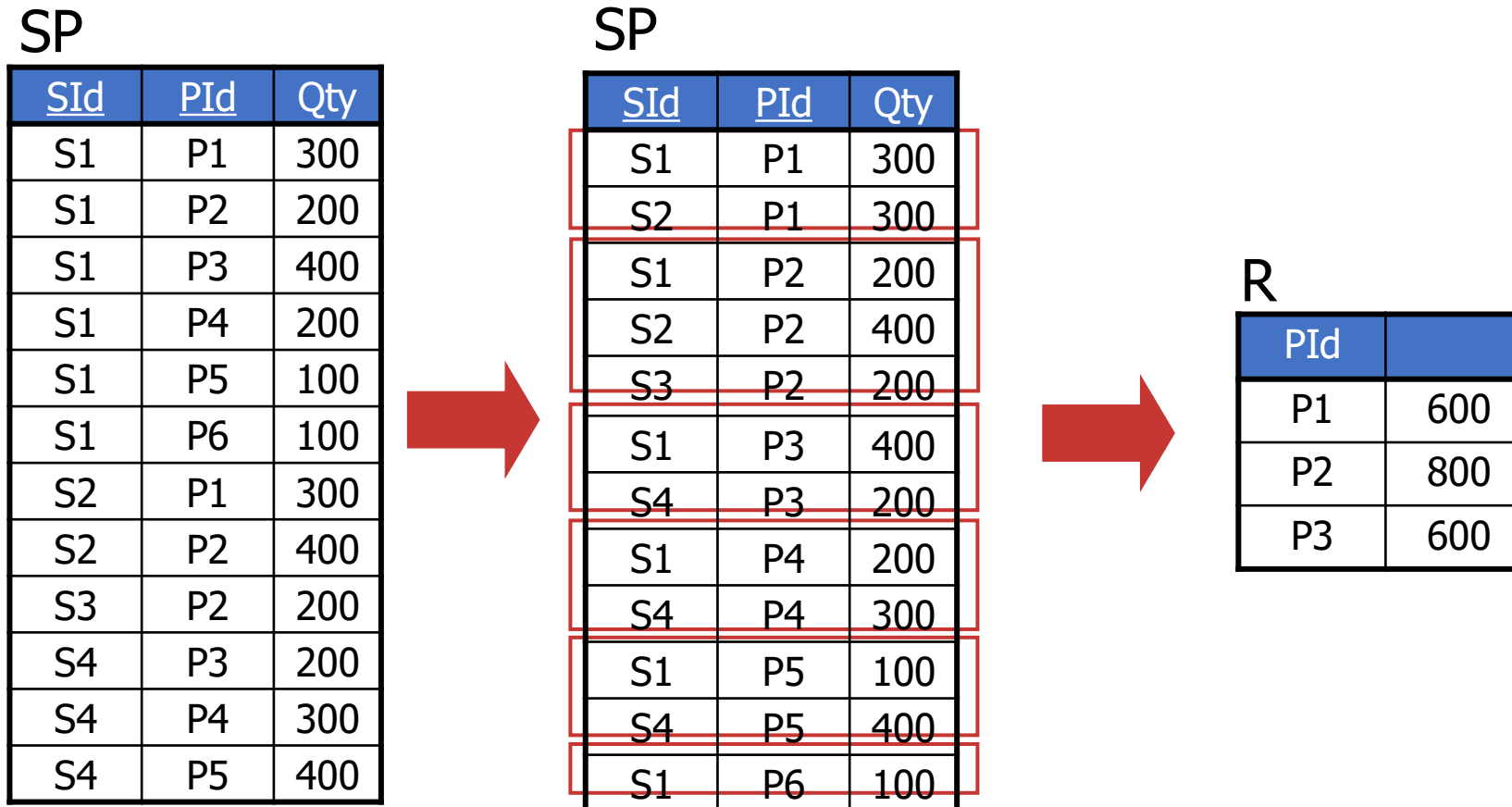
- You cannot use the WHERE clause to define selection conditions on groups
- Selection condition on groups expressed in HAVING clause:

## HAVING Group Conditions

- it is possible to specify conditions **only** on aggregated functions

# Group selection condition(n.1)

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



# Group selection condition (n.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 (n.2)

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

P

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

SP

<u>SId</u>	<u>PId</u>	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(n.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 (n.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.Colore	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