Query Language and Groupings

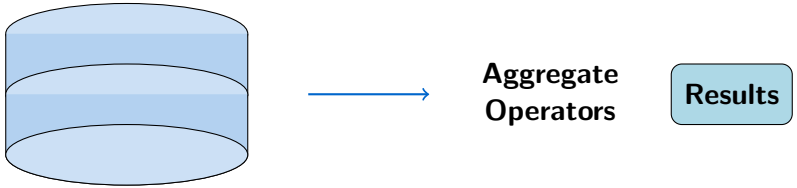
Aggregate Operators and SQL Clauses

Computer Science - IIS Fermi Sacconi Ceci

Ascoli Piceno

All rights reserved

Table of Contents



 All rights reserved

Aggregate Operators

SQL Extension

Compared to relational algebra, the most important extension introduced by SQL is that of **aggregate operators**.

Main characteristics:

- Aggregate multiple table rows
- Perform operations at *relation* level, not tuple level
- Return a **single aggregate value**
- Do not select a subset of rows, but calculate values

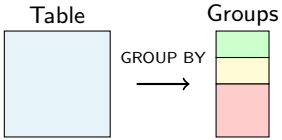
All rights reserved

Aggregate Operators

Functionality

Aggregation operations allow you to:

- **Group** tuples
- Perform **specific calculations** (sums, counts, statistics)
- **Divide** the table into subsets
- Group tuples with **same values** for specific attributes



All rights reserved

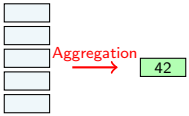
Aggregate Operators

Fundamental difference

Aggregation operators, unlike mathematical operators that act only on the tuple being processed:

Can be applied to multiple tuples of the table and are characterized by:

- Returning a **unique value**
- Operating on a **group of values**
- Working on values that form a **column**



All rights reserved

Aggregate Operators

Main aggregation operators

`AVG(field)` Calculates the **arithmetic mean**

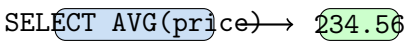
`COUNT(expr|*)` **Counts** rows

`MAX(expr)` Calculates the **maximum** value

`MIN(expr)` Calculates the **minimum** value

`SUM(field)` Calculates the **total sum**

`STDDEV(field)` Calculates the **standard deviation**



All rights reserved

Aggregate Operators

GROUP BY Clause

Specifies which fields to perform groupings on.

Important

Aggregate operators can only appear after these clauses:

- `SELECT`
- `HAVING`

```
SELECT category, COUNT(*) AS number
FROM products
GROUP BY category;
```


All rights reserved

The HAVING Clause

How it works

The `HAVING` clause allows you to insert a constraint on the data resulting from the `GROUP BY` grouping operation.

Difference with WHERE:

- `WHERE` operates on **database fields** (before grouping)
- `HAVING` operates on **fields resulting from groupings** (after)

```
SELECT department, AVG(salary) AS average
FROM employees
GROUP BY department
HAVING AVG(salary) > 30000;
```

 All rights reserved

COUNT Operator

Syntax

```
COUNT(expression | *)
```

```
SELECT COUNT(*) AS total_cars
FROM cars;
```

total_cars
12

Note: `COUNT(*)` counts all rows, including duplicates.

All rights reserved

COUNT Operator

Result

The result is **12**: all records that make up the table and are not NULL are counted, regardless of their content.

Important

`COUNT` is the only aggregation operator that considers null values in the calculation.

To count distinct values:

```
SELECT COUNT(DISTINCT brand) AS
   different_brands
FROM cars;
```

different_brands
8

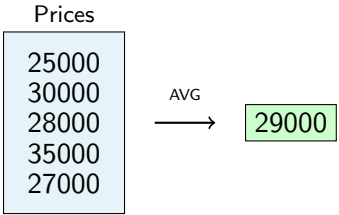
 All rights reserved

AVG Operator

Arithmetic Mean

Calculates the average of values in a numeric column.

```sql
SELECT AVG(price) AS average_price
FROM cars;
```



All rights reserved

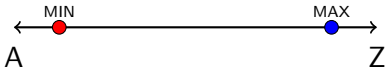
MIN and MAX Operators

How they work

`MAX` and `MIN` return the highest and lowest values contained within a column.

Behavior by data type:

- **Numeric fields:** returns the numeric value
- **Text fields:** identifies the field that, in alphabetical order, is last (MAX) or first (MIN)
- Ordering according to the **ASCII** code table
- **Lexicographic** ordering



All rights reserved

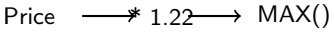
MIN and MAX Operators

Mathematical expressions

It is possible to use mathematical expressions as arguments for the `MIN` and `MAX` operators.

```
SELECT MAX(price * 1.22) AS max_price_vat
FROM cars;
```

```
SELECT MIN(price - discount) AS
   min_price_discounted
FROM cars;
```



All rights reserved

MIN and MAX Operators

```sql
SELECT MIN(salary) AS minimum_salary
FROM employees;
```

Name	Salary
Mario Rossi	28000
Laura Bianchi	32000
Giuseppe Verdi	**25000**
Anna Neri	35000

minimum_salary
25000

All rights reserved

MIN and MAX Operators

```sql
SELECT MAX(salary) AS maximum_salary
FROM employees;
```

Name	Salary
Mario Rossi	28000
Laura Bianchi	32000
Giuseppe Verdi	25000
Anna Neri	**35000**

maximum_salary
35000

All rights reserved

MIN and MAX Operators

```sql
SELECT MAX(surname) AS max_surname
FROM employees;
```

Surname
Bianchi
Neri
Rossi
Verdi

max_surname
Verdi

Alphabetical order: B ¡ N ¡ R ¡ V

All rights reserved

MIN and MAX Operators

```sql
SELECT MIN(surname) AS min_surname
FROM employees;
```

Surname
Bianchi
Neri
Rossi
Verdi

min_surname
Bianchi

Alphabetical order: B ¡ N ¡ R ¡ V

All rights reserved

MIN and MAX Operators

Complex expressions

Let's see how to apply arithmetic operators in the query. To know the difference between minimum and maximum employee salaries, expressed as a percentage:

```
SELECT
(MAX(salary) - MIN(salary)) /
MAX(salary) * 100 AS percentage_diff
FROM employees;
```

percentage_diff
28.57

The difference between maximum and minimum salary is 28.57%

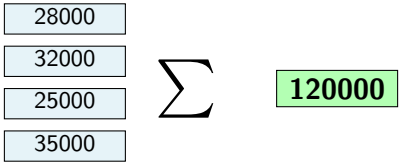
All rights reserved

SUM Operator

Total sum

Calculates the sum of all values in a numeric column.

```sql
SELECT SUM(salary) AS total_salaries
FROM employees;
```



All rights reserved

SUM Operator

```sql
SELECT
SUM(amount) AS total_sales,
SUM(quantity) AS total_pieces
FROM sales
WHERE year = 2024;
```

total_sales	total_pieces
158750.50	1243

Note

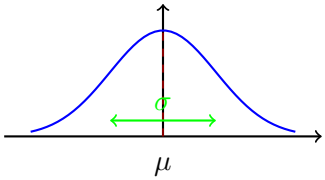
`SUM` ignores NULL values in the calculation.

All rights reserved

AVG and STDDEV Operators

```sql
SELECT
AVG(salary) AS average_salary,
STDDEV(salary) AS standard_deviation
FROM employees;
```

average_salary	standard_deviation
30000.00	4082.48



All rights reserved

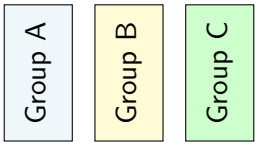
The GROUP BY Clause

Function

The `GROUP BY` clause is used to group and process uniformly different rows that have equal values in a specific column in the source table.

Syntax:

```sql
SELECT column1, aggregate_function(column2)
FROM table
WHERE condition
GROUP BY column1;
```



All rights reserved

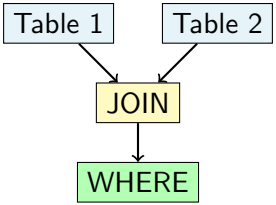
The GROUP BY Clause

Execution order

Taking into account the following considerations, let's clarify the priorities during the execution of a selection with grouping.

Processing phases:

1. Execution of the `JOIN`, if present
2. Selection on tuples with the condition after `WHERE`

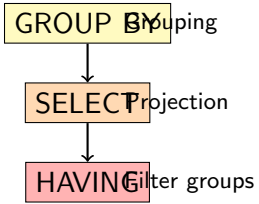


All rights reserved

The GROUP BY Clause

Processing phases (continued):

3. Grouping based on fields specified after `GROUP BY`
4. List of values specified after `SELECT` (target list)
5. Selection of tuples that satisfy the condition after `HAVING`



 All rights reserved

The GROUP BY Clause

```sql
SELECT department, COUNT(*) AS
    number_employees
FROM employees
GROUP BY department;
```

Original table:

Name	Department
Mario	Sales
Laura	IT
Giuseppe	Sales
Anna	IT
Carlo	Marketing

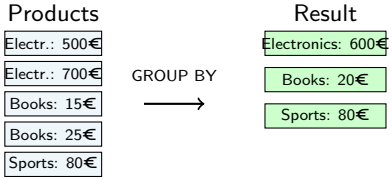
Result:

department	number_employees
Sales	2
IT	2
Marketing	1

All rights reserved

The GROUP BY Clause

```sql
SELECT category, AVG(price) AS average_price
FROM products
GROUP BY category;
```



All rights reserved

The GROUP BY Clause

```sql
SELECT
year,
month,
SUM(sales) AS total_sales
FROM monthly_sales
GROUP BY year, month
ORDER BY year, month;
```

year	month	total_sales
2023	1	45000
2023	2	52000
2024	1	48000
2024	2	55000

Grouping on **multiple columns**

All rights reserved

The GROUP BY Clause

```sql
SELECT
city,
COUNT(*) AS number_customers,
SUM(orders) AS total_orders
FROM customers
GROUP BY city;
```

city	number_customers	total_orders
Rome	15	234
Milan	22	387
Naples	8	145
Turin	12	198

All rights reserved

The GROUP BY Clause

```sql
SELECT
brand,
model,
COUNT(*) AS units_sold,
AVG(price) AS average_price
FROM cars_sold
GROUP BY brand, model
ORDER BY units_sold DESC;
```

brand	model	units	price
Fiat	Panda	45	12500
Volkswagen	Golf	38	22000
Ford	Fiesta	32	15000

All rights reserved

The GROUP BY Clause

```sql
SELECT
YEAR(sale_date) AS year,
MONTH(sale_date) AS month,
COUNT(*) AS number_transactions,
SUM(amount) AS revenue
FROM transactions
GROUP BY YEAR(sale_date), MONTH(sale_date);
```

year	month	transactions	revenue
2024	10	152	45200.00
2024	11	178	52800.00
2024	12	201	68500.00

Grouping with **functions** in columns

All rights reserved

The HAVING Clause

```sql
SELECT department, AVG(salary) AS
    average_salary
FROM employees
GROUP BY department
HAVING AVG(salary) > 30000;
```

All groups:

department	average_salary
Sales	28000
IT	35000
Marketing	27000
Production	32000

Result with HAVING:

department	average_salary
IT	35000
Production	32000

All rights reserved

The HAVING Clause

```sql
SELECT category, COUNT(*) AS number_products
FROM products
GROUP BY category
HAVING COUNT(*) >= 5;
```

category	number_products
Electronics	12
Clothing	8
Home	6

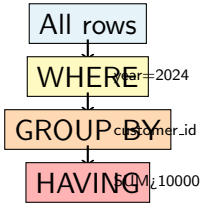
Note

`HAVING` filters groups **after** aggregation, while `WHERE` filters rows **before**.

All rights reserved

The HAVING Clause

```sql
SELECT
customer_id,
SUM(amount) AS total_purchases
FROM orders
WHERE year = 2024
GROUP BY customer_id
HAVING SUM(amount) > 10000;
```



All rights reserved

The HAVING Clause

```sql
SELECT
city,
COUNT(*) AS number_orders,
AVG(amount) AS average_amount
FROM orders
GROUP BY city
HAVING COUNT(*) > 10 AND AVG(amount) > 500;
```

city	number_orders	average_amount
Milan	25	675.50
Rome	18	820.30

Multiple conditions in HAVING

All rights reserved

The HAVING Clause

```
SELECT
brand,
COUNT(*) AS models,
MIN(price) AS min_price,
MAX(price) AS max_price
FROM automobiles
GROUP BY brand
HAVING MAX(price) - MIN(price) > 20000;
```

brand	models	min_price	max_price
BMW	8	35000	85000
Mercedes	6	40000	95000

HAVING with **expressions** between aggregations

All rights reserved

Limiting Result Tuples

LIMIT Clause

Allows you to limit the number of rows returned by the query.

Syntax:

```
SELECT columns
FROM table
WHERE condition
ORDER BY column
LIMIT number;
```

Common uses:

- Display only the first N results
- Implement pagination
- Optimize query performance

All rights reserved

Limiting Result Tuples

```sql
SELECT name, surname, salary
FROM employees
ORDER BY salary DESC
LIMIT 5;
```

Top 5 highest salaries:

name	surname	salary
Anna	Neri	45000
Mario	Rossi	42000
Laura	Bianchi	38000
Giuseppe	Verdi	35000
Carlo	Gialli	33000

Returns only the first 5 ordered rows

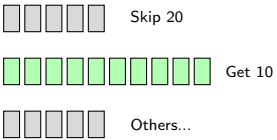
All rights reserved

Limiting Result Tuples

Pagination with OFFSET

`OFFSET` allows you to skip a specified number of rows.

```sql
SELECT name, price
FROM products
ORDER BY price DESC
LIMIT 10 OFFSET 20;
```



Useful for pagination: page 3 = LIMIT 10 OFFSET 20

All rights reserved

Limiting Result Tuples

```sql
SELECT
product,
sales,
revenue
FROM sales_statistics
ORDER BY revenue DESC
LIMIT 3;
```

Top 3 products by revenue:

product	sales	revenue
Laptop Pro	234	234000 €
Smartphone X	456	182400 €
Tablet Plus	198	99000 €

Best Practice

Always use `ORDER BY` with `LIMIT` for consistent results.

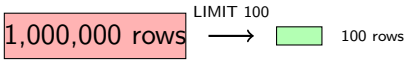
All rights reserved

Limiting Result Tuples

Advantages of LIMIT

- **Performance:** reduces database load
- **Memory:** transfers less data
- **Usability:** improves user experience
- **Testing:** useful for testing queries on large datasets

```sql
-- Test query on large table
SELECT * FROM event_logs
WHERE date >= '2024-01-01'
LIMIT 100;
```



All rights reserved