



GROUP 9

**IQUEN L MARBA
ORLY ORGE
WENDEL TUPAZ**





MySQL CROSS JOIN

MySQL is an open-source relational database management system that uses Structured Query Language (SQL) to manipulate databases. It stores data in a table format. It provides various statements to perform Create, Read, Update, and Delete operations on a database table. Among these operations, MySQL also provides the CROSS JOIN statement to combine rows from different tables.



MySQL CROSS JOIN

MySQL CROSS JOIN is a join procedure that is used to combine two or more tables. It is also known as a Cartesian join and returns the Cartesian product of two or more tables. It combines each row of one table with each row of another table and returns a new table with all possible combinations.

```
MariaDB [shop]> select * from customer;
```

id	fname	date
1	iquen	2024-03-21 23:10:03
2	von	2024-03-21 23:10:03
3	wendel	2024-03-21 23:10:03
4	only	2024-03-21 23:10:03

4 rows in set (0.001 sec)

```
MariaDB [shop]>
```

```
MariaDB [shop]> insert into item(item_id,itemname)values(2,'snipe')
-> ,(3,'knife')
-> ,(4,'granade');
```

Query OK, 3 rows affected (0.004 sec)

Records: 3 Duplicates: 0 Warnings: 0

```
MariaDB [shop]> select *from item
```

```
-> ;
```

item_id	itemname	date
1	gun	2024-03-21 23:15:20
2	snipe	2024-03-21 23:16:33
3	knife	2024-03-21 23:16:33
4	granade	2024-03-21 23:16:33

4 rows in set (0.001 sec)


```
MariaDB [shop]> select * from customer cross join item;
```

id	fname	date	item_id	itemname	date
1	iquen	2024-03-21 23:10:03	1	gun	2024-03-21 23:15:20
2	von	2024-03-21 23:10:03	1	gun	2024-03-21 23:15:20
3	wendel	2024-03-21 23:10:03	1	gun	2024-03-21 23:15:20
4	only	2024-03-21 23:10:03	1	gun	2024-03-21 23:15:20
1	iquen	2024-03-21 23:10:03	2	snipe	2024-03-21 23:16:33
2	von	2024-03-21 23:10:03	2	snipe	2024-03-21 23:16:33
3	wendel	2024-03-21 23:10:03	2	snipe	2024-03-21 23:16:33
4	only	2024-03-21 23:10:03	2	snipe	2024-03-21 23:16:33
1	iquen	2024-03-21 23:10:03	3	knife	2024-03-21 23:16:33
2	von	2024-03-21 23:10:03	3	knife	2024-03-21 23:16:33
3	wendel	2024-03-21 23:10:03	3	knife	2024-03-21 23:16:33
4	only	2024-03-21 23:10:03	3	knife	2024-03-21 23:16:33
1	iquen	2024-03-21 23:10:03	4	granade	2024-03-21 23:16:33
2	von	2024-03-21 23:10:03	4	granade	2024-03-21 23:16:33
3	wendel	2024-03-21 23:10:03	4	granade	2024-03-21 23:16:33
4	only	2024-03-21 23:10:03	4	granade	2024-03-21 23:16:33

16 rows in set (0.001 sec)

The HAVING clause is used to filter grouped data in SQL queries.

- It is applied after the GROUP BY clause and before the ORDER BY clause.
- HAVING is used with aggregate functions (e.g., 'SUM', 'COUNT', 'AVG', 'MAX', 'MIN') to filter grouped results.

```
SELECT department_id, SUM(salary) AS total_salary  
FROM employees  
GROUP BY department_id  
HAVING SUM(salary) > 100000;
```

This query retrieves department IDs and their total salaries, filtering out departments where the total salary exceeds \$100,000.

```
SELECT category, COUNT(*) AS num_products  
FROM products  
GROUP BY category  
HAVING COUNT(*) > 10;
```

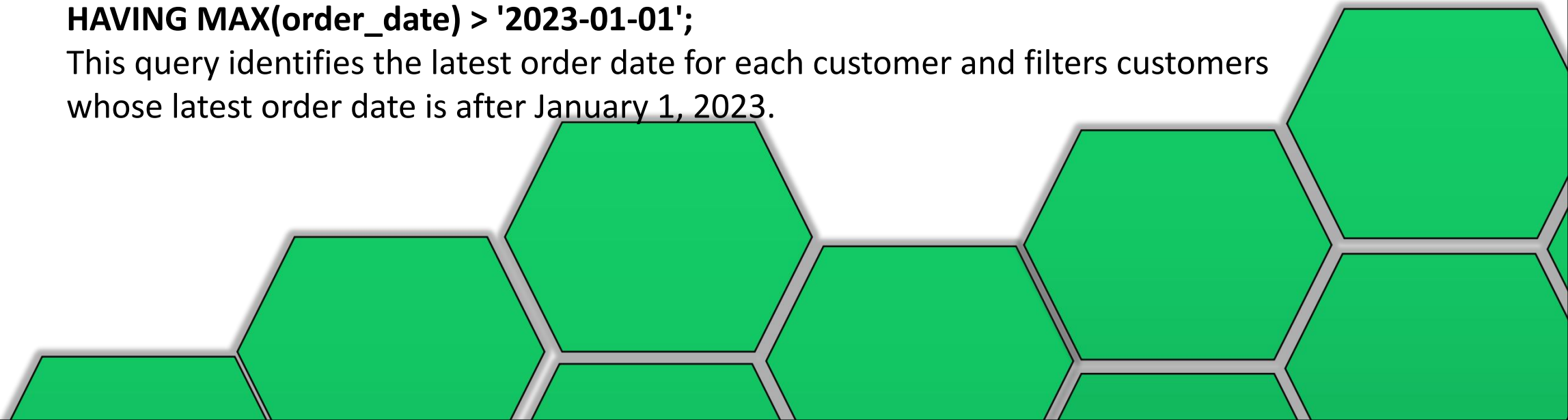
This query counts the number of products in each category, filtering categories with more than 10 products.

```
SELECT supplier_id, AVG(unit_price) AS avg_price  
FROM products  
GROUP BY supplier_id  
HAVING AVG(unit_price) > 50;
```

This query calculates the average unit price by supplier and filters out suppliers whose average unit price is more than \$50.

```
SELECT customer_id, MAX(order_date) AS last_order_date  
FROM orders  
GROUP BY customer_id  
HAVING MAX(order_date) > '2023-01-01';
```

This query identifies the latest order date for each customer and filters customers whose latest order date is after January 1, 2023.



The **GROUP BY** statement groups rows that have the same values into summary rows

The **GROUP BY** statement is often used with aggregate functions (**COUNT()**, **MAX()**, **MIN()**, **SUM()**, **AVG()**) to group the result-set by one or more columns.

