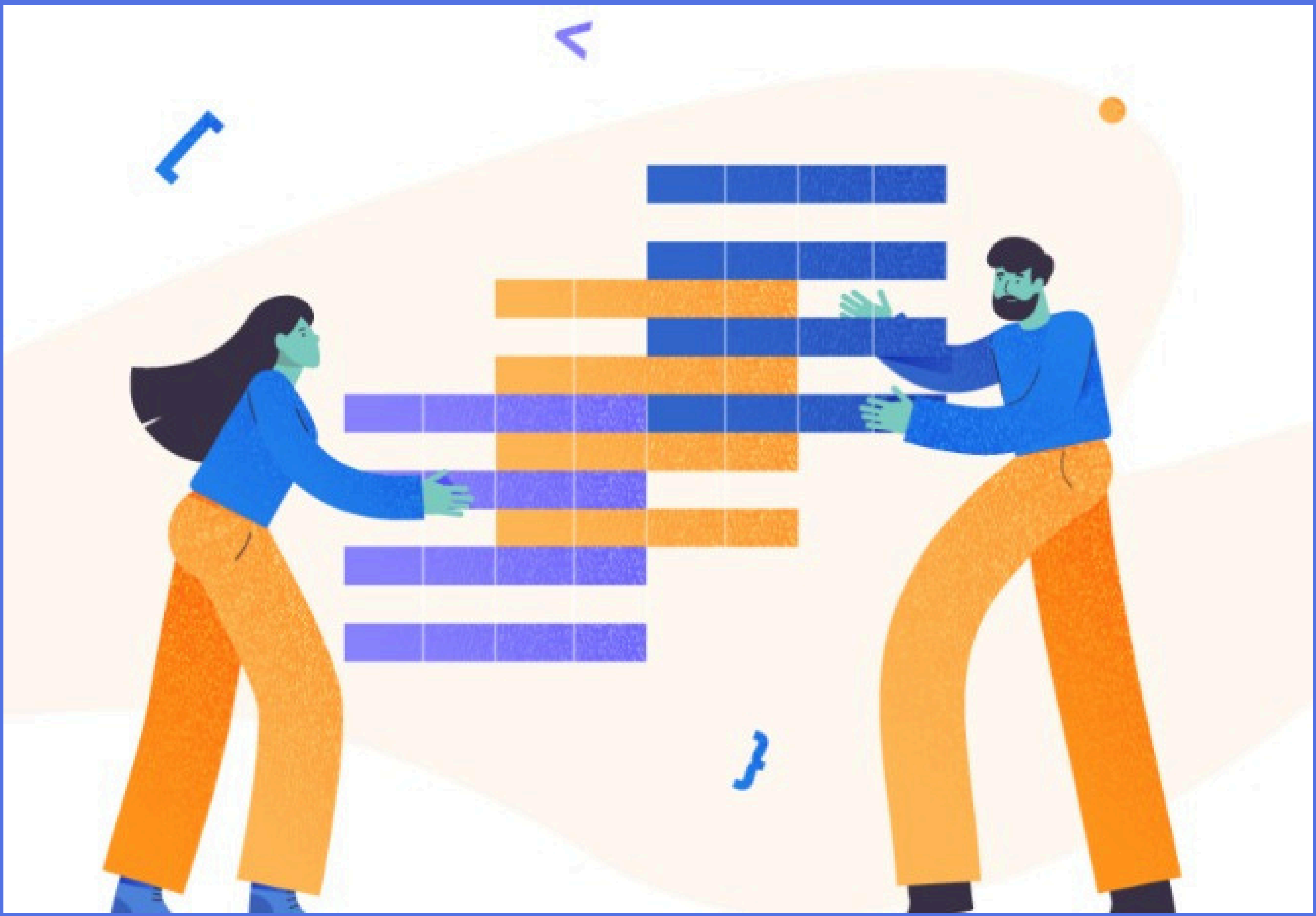
Simple Guide to Aggregate Functions and GROUP BY in MySQL



Introduction to Aggregate Functions

Aggregate functions in SQL perform calculations on multiple rows of data and return a single summary value. These are essential for data analysis, allowing you to calculate totals, averages, and other key metrics from large datasets.

Key Functions:
SUM(): Adds up all the values in a numeric column.
COUNT(): Returns the number of rows.
AVG(): Calculates the average value in a numeric column.
MIN(): Returns the smallest value.
MAX(): Returns the largest value.

Example of Aggregate Functions

Query:

```
SELECT COUNT(*) AS TotalSales FROM Sales;
```

Returns the total number of sales records.

Query:

```
SELECT SUM(Quantity) AS TotalQuantity
FROM Sales;
```

Returns the total quantity of all products sold.

Query:

```
SELECT AVG(Price) AS AveragePrice FROM
Sales;
```

Returns the average price of products sold.

Query:

```
SELECT MAX(Price) AS MaxPrice, MIN(Price)
AS MinPrice FROM Sales;
```

Returns the highest and lowest product prices.

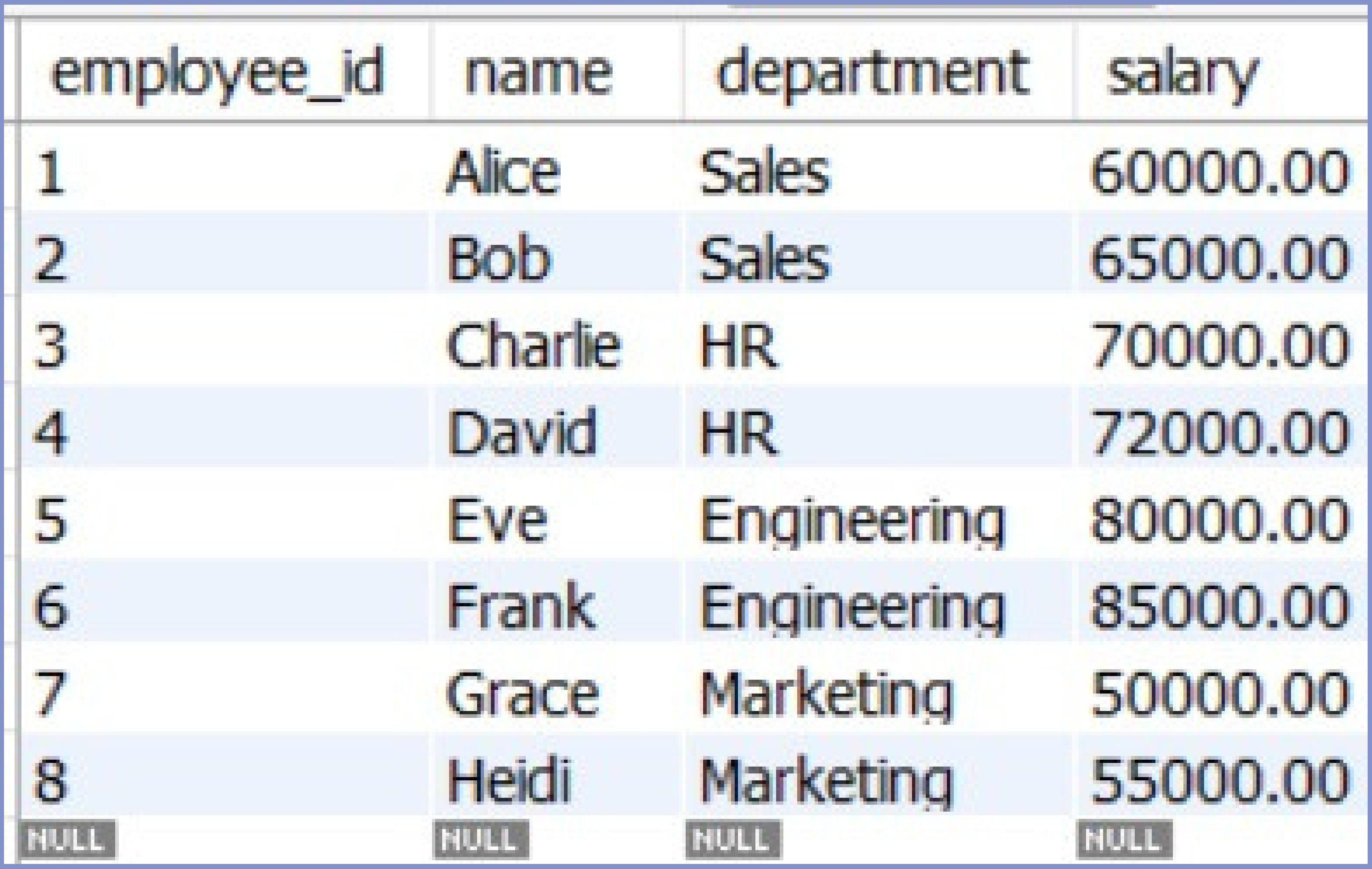
Introduction to the GROUP BY Clause

The GROUP BY clause is used to arrange identical data into groups. It's a powerful SQL tool that works with aggregate functions to produce summarized data for each group.

Without GROUP BY, aggregate functions like SUM() or COUNT() will calculate values for the entire dataset. With GROUP BY, you can group data by specific categories and calculate results for each group.

Example of GROUP BY with Aggregate Functions

Consider the following table to consider the example

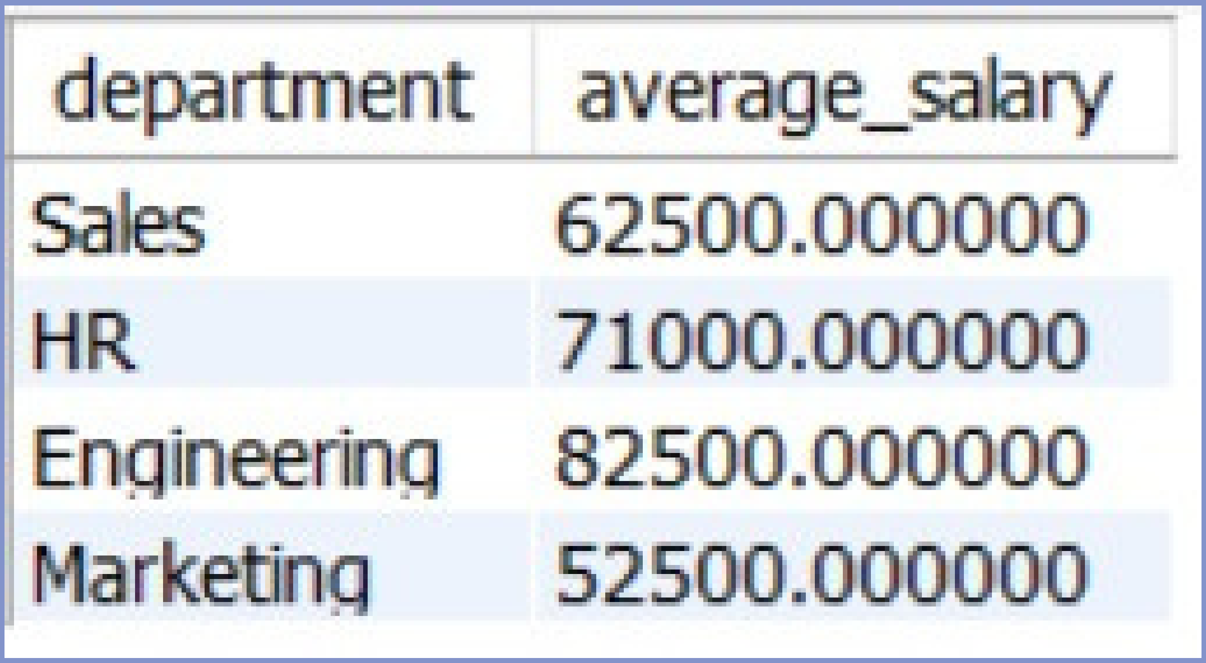


employee_id	name	department	salary
1	Alice	Sales	60000.00
2	Bob	Sales	65000.00
3	Charlie	HR	70000.00
4	David	HR	72000.00
5	Eve	Engineering	80000.00
6	Frank	Engineering	85000.00
7	Grace	Marketing	50000.00
8	Heidi	Marketing	55000.00
NULL	NULL	NULL	NULL

Query:

```
SELECT department, AVG(salary) AS
average_salary FROM employees
GROUP BY department;
```

This query groups employees by their department and calculates the average salary for each group using the AVG() function.



department	average_salary
Sales	62500.000000
HR	71000.000000
Engineering	82500.000000
Marketing	52500.000000

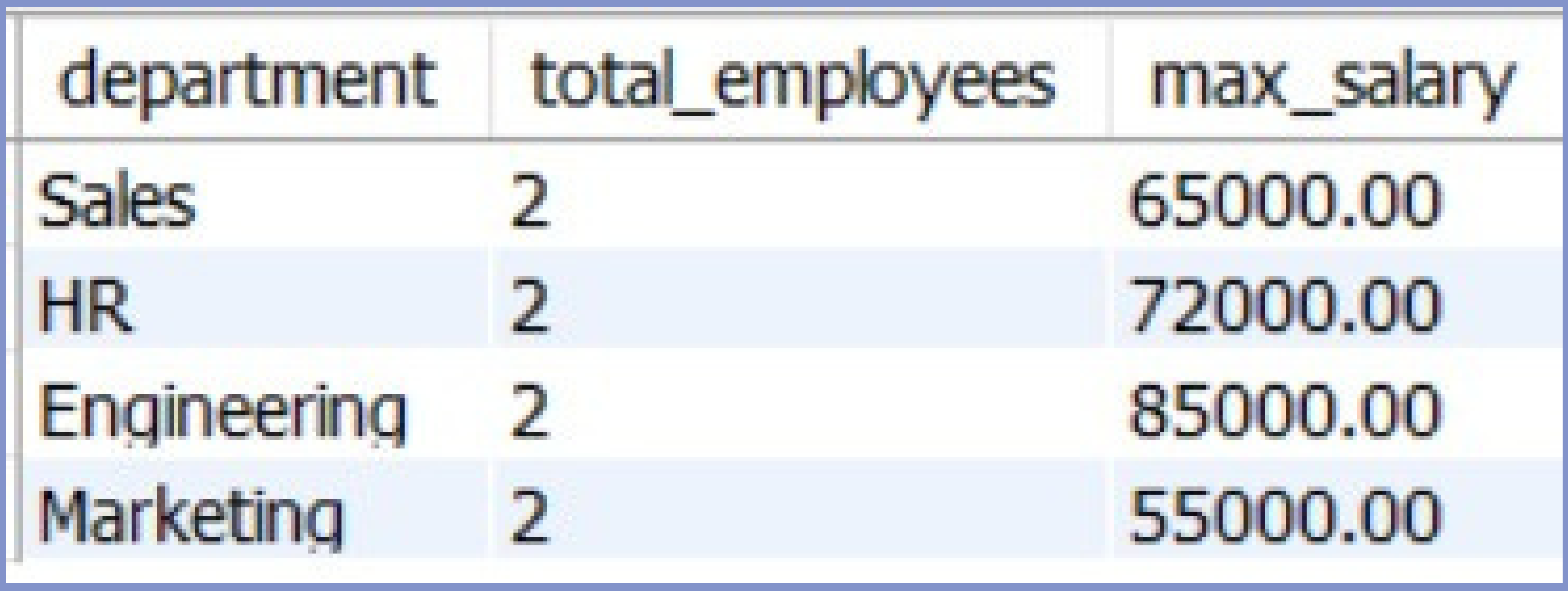
Combining Multiple Aggregate Functions with GROUP BY

Query:

SELECT department, COUNT(employee_id) AS total_employees, MAX(salary) AS max_salary FROM employees GROUP BY department;

Explanation:

In this example, the query groups employees by department and then applies two aggregate functions COUNT and MAX.



department	total_employees	max_salary
Sales	2	65000.00
HR	2	72000.00
Engineering	2	85000.00
Marketing	2	55000.00

Handling NULLs in Aggregate Functions

When working with aggregate functions, it’s important to understand how NULL values are treated. Most aggregate functions (except COUNT()) ignore NULL values. This can impact results if your dataset has missing data.
Example query:

```
SELECT department, SUM(salary) AS
total_salary FROM employees
GROUP BY department;
```

In this case, SUM(salary) will ignore any rows where salary is NULL, which can lead to skewed results if you don’t account for missing data.
Tip: Always check for NULLs before using aggregate functions

Practical Use Cases of Aggregate Functions and GROUP BY

Sales Analysis: Calculate the total sales per product or region.
Customer Insights: Find the average number of purchases per customer.
HR Analysis: Track the maximum, minimum, or average salaries per department.
Inventory Management: Determine the total quantity of products sold.

“The best way to master SQL aggregate functions and the GROUP BY clause is through consistent practice! Try different datasets and explore combining multiple functions to uncover hidden insights in your data.”