SQL GROUP BY

The `GROUP BY` clause is used to group rows that have the same values in specified columns and perform aggregate functions (e.g., `COUNT`, `SUM`, `AVG`, `MIN`, `MAX`) on each group.

Key Features

1. **Purpose:** Group data based on one or more columns.
2. **Works With Aggregate Functions:** Used alongside `COUNT`, `SUM`, `AVG`, etc.
3. **Order of Execution:** The `GROUP BY` clause follows the `WHERE` clause but precedes the `HAVING` and `ORDER BY` clauses.
4. **Syntax:**

```sql
SELECT column1, column2, AGGREGATE_FUNCTION(column3)
FROM table_name
WHERE condition
GROUP BY column1, column2
HAVING condition
ORDER BY column1;
```

Basic Example

Count employees by department:

```sql
SELECT department_id, COUNT(employee_id) AS total_employees
FROM employees
GROUP BY department_id;
```

Result:

	department_id	total_employees
1	1	10
2	2	8
3	3	5

+ Neu

Using Multiple Columns in GROUP BY

You can group by more than one column.

Example: Count employees by department and job title:

```sql
SELECT department_id, job_title, COUNT(employee_id) AS total_employees
FROM employees
```

```
GROUP BY department_id, job_title;
```

Result:

	department_id	job_title	total_employees
1	1	Manager	2
2	1	Engineer	8
3	2	Analyst	5
4	2	Clerk	3

+ Neu

Using GROUP BY with Aggregate Functions

1. **SUM:** Total salary by department.

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

2. **AVG:** Average salary by job title.

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

3. **MIN/MAX:** Minimum and maximum salary by department.

```sql
SELECT department_id, MIN(salary) AS min_salary, MAX(salary) AS max_salary
FROM employees
GROUP BY department_id;
```

Using GROUP BY with WHERE Clause

The `WHERE` clause filters rows before grouping.

Example: Total salary for departments with salaries over 30,000:

```sql
SELECT department_id, SUM(salary) AS total_salary
FROM employees
WHERE salary > 30000
GROUP BY department_id;
```

Using GROUP BY with HAVING Clause

The `HAVING` clause filters groups after aggregation (similar to `WHERE` but for aggregated results).

Example: Departments with total salary > 100,000:

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

GROUP BY with ORDER BY

You can sort the grouped results using `ORDER BY`.

Example: Departments sorted by total salary:

```sql
SELECT department_id, SUM(salary) AS total_salary
FROM employees
GROUP BY department_id
ORDER BY total_salary DESC;
```

GROUP BY with Aliases

You can use column aliases for readability.

Example:

```sql
SELECT department_id AS dept, COUNT(employee_id) AS employees
FROM employees
GROUP BY department_id;
```

GROUP BY with CASE

You can use `CASE` statements within the `GROUP BY` clause.

Example: Categorize employees by salary ranges:

```sql
SELECT
  CASE
    WHEN salary < 30000 THEN 'Low'
    WHEN salary BETWEEN 30000 AND 70000 THEN 'Medium'
    ELSE 'High'
  END AS salary_category,
  COUNT(employee_id) AS total_employees
FROM employees
GROUP BY
  CASE
    WHEN salary < 30000 THEN 'Low'
    WHEN salary BETWEEN 30000 AND 70000 THEN 'Medium'
    ELSE 'High'
  END;
```

Result:

	salary_category	total_employees
1	Low	15
2	Medium	20
3	High	5

Neu

GROUP BY Best Practices

1. Use only the columns you want to group by or aggregate in the `SELECT` clause.
2. Always test your query without `GROUP BY` first to understand the raw data.
3. Combine `HAVING` with aggregate functions to filter grouped results.

Would you like to try some practice exercises or need help with a specific query?