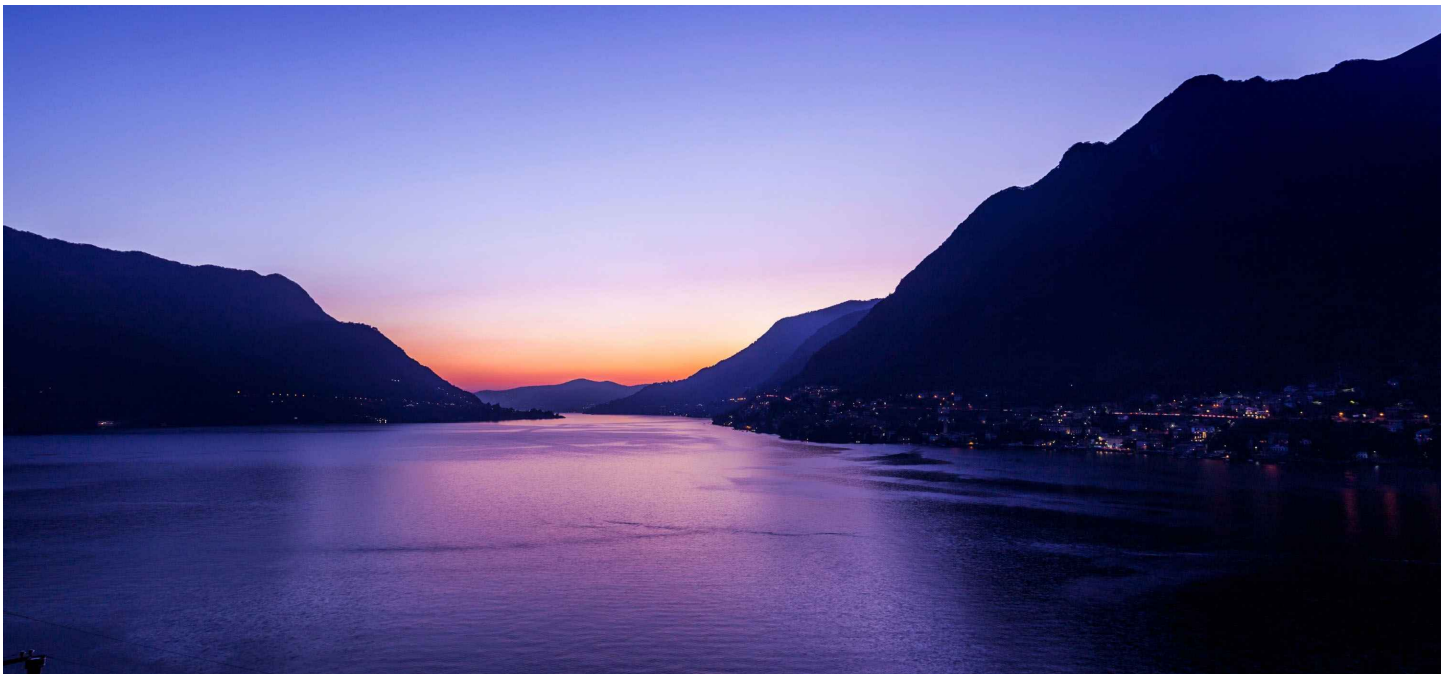
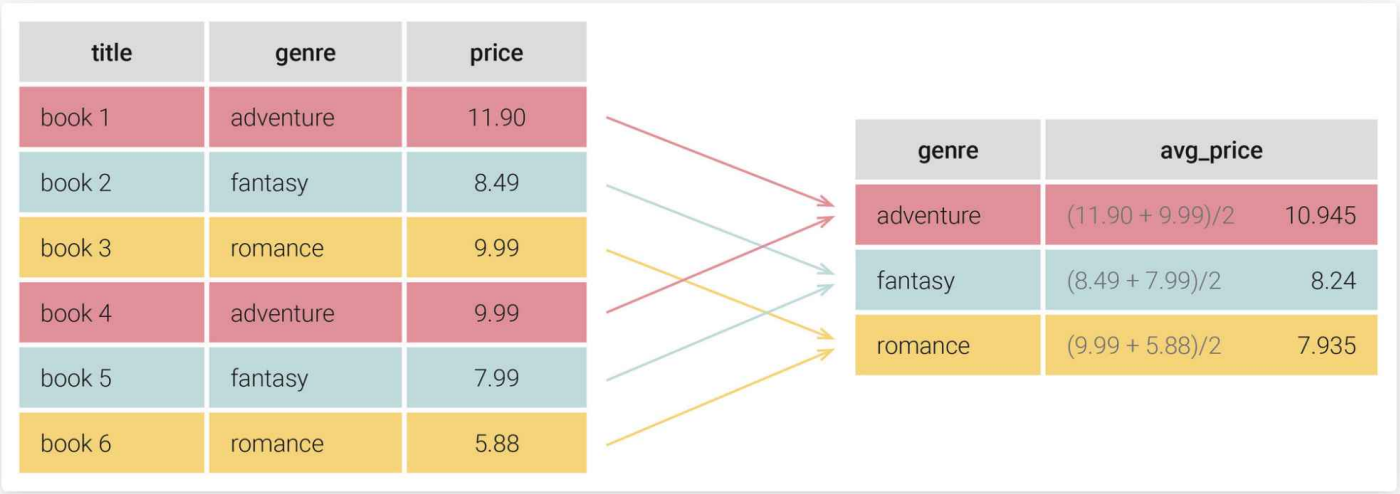


7-SQL Group by & Having

Author: Vincent Lau

Learning Objectives

- Understand the basic syntax and result of using GROUP BY.
- Understand the relationship between Aggregation Function and GROUP BY.
- Why do we need to use Aggregation Functions when we try to select the field other than the fields grouping by?
- Understand the result of using or not using HAVING.
- What is the difference between using HAVING and WHERE.



Introduction

The `GROUP BY` clause in SQL is used to group rows that have the same values in specified columns into summary rows. It's commonly used with aggregate functions (like `SUM`, `COUNT`, `AVG`, etc.) to perform calculations on grouped data.

Let's create a simple table called `sales` with columns for `product`, `region`, and `sales_amount`.

```sql
CREATE TABLE sales (
    product VARCHAR(50),
    region VARCHAR(50),
    sales_amount DECIMAL(10, 2)
);

INSERT INTO sales (product, region, sales_amount) VALUES
    ('Product A', 'North', 1000.00),
    ('Product B', 'North', 1500.00),
    ('Product A', 'South', 800.00),
    ('Product B', 'South', 1200.00),
    ('Product A', 'East', 1300.00);
```

GROUP BY & HAVING

Simple Query Without `GROUP BY`

Before we use `GROUP BY`, let's run a simple query to see all the sales data without any grouping. This query retrieves all rows from the `sales` table and displays the raw data.

```sql
SELECT * FROM sales;
```

The result will look like this:

```
| product | region | sales_amount |
|---------/--------/------------/
| Product A | North  | 1000.00    |
| Product B | North  | 1500.00    |
| Product A | South  | 800.00    |
| Product B | South  | 1200.00    |
| Product A | East   | 1300.00    |
```

Basic `GROUP BY` Example

Now, let's use `GROUP BY` to group the data by `region` and calculate the total sales amount for each region.

```sql
SELECT region, SUM(sales_amount) AS total_sales
FROM sales
GROUP BY region;
```

This query groups the sales data by the `region` column and calculates the total sales amount for each region. The result will look like this:

```
| region | total_sales |
|--------/------------/
| North  | 2500.00    |
| South  | 2000.00    |
| East   | 1300.00    |
```

`GROUP BY` with Multiple Columns

You can also use `GROUP BY` with multiple columns to create more detailed groups. Let's group by both `product` and `region`.

```sql
SELECT product, region, SUM(sales_amount) AS total_sales
FROM sales
GROUP BY product, region;
```

This query groups the data by both `product` and `region` columns and calculates the total sales amount for each product in each region. The result will show sales totals for every product in every region.

The result will look like this:

```
| product | region | total_sales |
|---------/--------/------------/
| Product A | North  | 1000.00    |
| Product B | North  | 1500.00    |
| Product A | South  | 800.00    |
| Product B | South  | 1200.00    |
| Product A | East   | 1300.00    |
```

Filtering Groups with `HAVING`

The `HAVING` clause in SQL is used to filter the results of a `GROUP BY` query based on the results of aggregate functions. It's often used in combination with the `GROUP BY` clause to filter grouped data. Let's understand `HAVING` and its differences from `WHERE` with examples:

You can **filter groups** using the `HAVING` clause.

For example, if you want to find regions with total sales >= 2000, you can use `HAVING` like this:

```sql
SELECT region, SUM(sales_amount) AS total_sales
FROM sales
GROUP BY region
HAVING SUM(sales_amount) >= 2000;
```

The result will look like this:

This query filters the groups using `HAVING` and displays only regions with total sales >= 2000.

So, the group 'East' was filtered.

```
| region | total_sales |
|--------/------------/
| North  | 2500.00    |
| South  | 2000.00    |
```

Combining `GROUP BY` with Other Clauses

You can combine `GROUP BY` with other SQL clauses like `WHERE` and `ORDER BY`. For example, you can find the region with the highest total sales:

```sql
SELECT region, SUM(sales_amount) AS total_sales
FROM sales
GROUP BY region
HAVING SUM(sales_amount) >= 2000
ORDER BY total_sales ASC
LIMIT 1;
```

The result will look like this:

```
| region | total_sales |
|--------/-------------/
| South  | 2000.00     |
```

This query combines `GROUP BY` with `ORDER BY` to sort regions by total sales in descending order and uses `LIMIT` to retrieve only the top result.

HAVING VS WHERE

1. `WHERE` Clause

- The `WHERE` clause **filters rows before they are grouped or aggregated**.
- It operates on individual rows in the original table.
- It cannot be used with aggregate functions.
- Typically used for row-level filtering.

2. `HAVING` Clause

- The `HAVING` clause **filters the whole groups after grouping**.
- It operates on aggregated values, typically on grouped rows.
- It can be used with aggregate functions (e.g., `SUM`, `COUNT`, `AVG`).
- Typically used for filtering grouped or summarized data.

Example - Using `WHERE` and `HAVING`

Let's say we have a table called `orders` with columns `customer_id` and `total_amount`. We want to find customers who have placed orders with a total amount greater than 1000.

Using `WHERE`:

```
SELECT customer_id, SUM(total_amount) AS total_purchase
FROM orders
WHERE total_amount > 1000
GROUP BY customer_id;
```

This query uses `WHERE` to filter individual rows before they are grouped. It calculates the total purchase amount for each customer and only includes customers whose total purchase is greater than 1000.

Using `HAVING`:

```
SELECT customer_id, SUM(total_amount) AS total_purchase
FROM orders
GROUP BY customer_id
HAVING SUM(total_amount) > 1000;
```

This query first groups rows by `customer_id`, calculates the total purchase amount for each customer using `SUM`, and then filters the grouped results using `HAVING`. It includes only customers whose total purchase (the result of the aggregate function) is greater than 1000.

In summary, the key difference is that `WHERE` filters rows before they are grouped, while `HAVING` filters grouped rows after aggregation. You use `WHERE` for row-level filtering and `HAVING` for filtering grouped or summarized data based on aggregate function results.