Chapter: Discovering SQL Trends

Topic: GROUP BY and HAVING Clause in SQL

Section 1: Learn

1.1 Introduction to GROUP BY and HAVING

As your data grows, raw records alone aren't enough. To make sense of large datasets and uncover trends, we use **aggregation**.

SQL offers two powerful clauses to help with this:

- GROUP BY — helps to **group rows** by one or more columns.
- HAVING — helps to **filter** these grouped results.

Together, they enable data summarization, trend analysis, and performance tracking.

1.2 GROUP BY Clause

The GROUP BY clause groups rows that have the **same values** in specified columns into summary rows.

Why use it?

Suppose you want to:

- Know the number of employees in each department.
- Calculate average marks per subject.
- Get total revenue by region.

All these require **grouping** and **aggregating** data.

Syntax:

```
SELECT column_name, AGGREGATE_FUNCTION(column_name)
FROM table_name
GROUP BY column_name;
```

Example 1: Count employees per department

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

Example 2: Group by multiple columns

```
SELECT department, gender, COUNT(*) AS count
FROM employees
GROUP BY department, gender;
```

> Groups employees first by department, then by gender.

1.3 Aggregate Functions with GROUP BY

GROUP BY is most useful when used with aggregate functions such as:

Function	Description
SUM()	Adds up values in a column
AVG()	Calculates average

Function	Description
COUNT()	Counts rows
MAX()	Finds the maximum value
MIN()	Finds the minimum value

Example:

```
SELECT category, SUM(sales) AS total_sales
FROM products
GROUP BY category;
```

> Groups products by category and shows the total sales per category.

1.4 HAVING Clause

The HAVING clause is used to **filter grouped data**. You cannot use WHERE on aggregate results — for that, you need HAVING.

Syntax:

```
SELECT column_name, AGGREGATE_FUNCTION(column_name)
FROM table_name
GROUP BY column_name
HAVING AGGREGATE_FUNCTION(column_name) condition;
```

Example:

```
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department
HAVING AVG(salary) > 50000;
```

> Shows only those departments where average salary exceeds ₹50,000.

1.5 Difference Between WHERE and HAVING

Feature	WHERE	HAVING
Use On	Raw (individual) rows	Aggregated (grouped) results
Used With	Any SELECT query	Must be used with GROUP BY or aggregates
Evaluates	Before GROUP BY	After GROUP BY

Example:

```
-- WHERE filters individual employees
SELECT * FROM employees
WHERE salary > 50000;


-- HAVING filters average salary of departments
SELECT department, AVG(salary)
FROM employees
GROUP BY department
```

```
HAVING AVG(salary) > 50000;
```

1.6 ORDER BY with GROUP BY

Once grouping and aggregation is done, we can **sort** the grouped results using ORDER BY.

```
SELECT department, COUNT(*) AS emp_count
FROM employees
GROUP BY department
ORDER BY emp_count DESC;
```

> Displays departments with highest to lowest employee count.

1.7 Real-World Use Cases

- **Retail:** Group sales by region, category, or time.
- **Education:** Group marks by class, gender, or subject.
- **HR:** Group employees by department, tenure, or role.
- **Finance:** Group transactions by month or by account type.

Section 2: Practise

Exercise 1: Students per Class

```
SELECT class, COUNT(*) AS student_count
FROM students
GROUP BY class;
```

Exercise 2: Average Marks by Subject

```
SELECT subject, AVG(marks) AS average_marks
FROM results
GROUP BY subject;
```

Exercise 3: Sales by Product with Total Revenue

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

Exercise 4: Customers with More than 3 Orders

```
SELECT customer_id, COUNT(*) AS order_count
FROM orders
GROUP BY customer_id
HAVING COUNT(*) > 3;
```

Exercise 5: Maximum Sales by Region

```
SELECT region, MAX(sales_amount) AS top_sale
FROM regional_sales
GROUP BY region;
```

Exercise 6: Departments with > 5 Employees

```
SELECT department, COUNT(*) AS total
FROM employees
GROUP BY department
HAVING COUNT(*) > 5;
```

Exercise 7: Cities with Avg Order Value > ₹1000

```
SELECT city, AVG(order_value) AS avg_order
FROM orders
GROUP BY city
HAVING AVG(order_value) > 1000;
```

Section 3: FAQ – Know More

Q1. Can I group by expressions or functions?

Yes. For example:

```
SELECT YEAR(join_date), COUNT(*)
FROM employees
GROUP BY YEAR(join_date);
```

Q2. Can I use GROUP BY on text columns?

Yes. You can group by strings like names, cities, regions, etc.

Q3. What happens if a column in SELECT is not in GROUP BY?

SQL will throw an error unless that column is part of an aggregate function or the database allows non-standard settings.

Q4. Can I group and filter by dates?

Yes. Use DATE(), YEAR(), MONTH() functions with GROUP BY and HAVING.

Q5. Can I nest aggregate functions like AVG(SUM())?

Not directly in SQL. You may need subqueries or CTEs to do layered aggregation.

End of Notes for Chapter: Discovering SQL Trends