🎯 Topic: GROUP BY & HAVING Clause

◈ 1.Concept

GROUP BY Clause

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

It is generally used with aggregate functions like SUM(), AVG(), COUNT(), MAX(), MIN().

It helps to generate summarized reports — e.g., department-wise, region-wise, product-wise data.

HAVING Clause

The HAVING clause filters the results after grouping.

It works like the WHERE clause, but it is used with aggregated data.

Example: "Show only departments where the average marks are above 75."

◈ 2️⃣ SQL Example (PostgreSQL)

```
Drop old table if exists
DROP TABLE IF EXISTS student;

Create table
CREATE TABLE student (
    student_id SERIAL PRIMARY KEY,
    student_name VARCHAR(50),
    dept_name VARCHAR(50),
    marks INT,
    fees NUMERIC(10,2)
);

Insert sample data
INSERT INTO student (student_name, dept_name, marks, fees) VALUES
('Ravi', 'CSE', 85, 35000),
('Sneha', 'CSE', 90, 36000),
('Arjun', 'ECE', 78, 34000),
('Deepa', 'ECE', 88, 33000),
('Kiran', 'EEE', 60, 32000),
('Lalitha', 'EEE', 72, 31000),
('Manoj', 'MECH', 55, 30000),
('Sujatha', 'MECH', 65, 31000);
```

==========

GROUP BY Examples

==========

1. Department-wise total students

```
SELECT dept_name, COUNT(*) AS total_students
FROM student
GROUP BY dept_name;
```

2. Department-wise average marks

```
SELECT dept_name, AVG(marks) AS avg_marks
FROM student
GROUP BY dept_name;
```

3. Department-wise total fees

```
SELECT dept_name, SUM(fees) AS total_fees
FROM student
GROUP BY dept_name;
```

==========

HAVING Clause Examples

==========

1. Departments where average marks > 75

```
SELECT dept_name, AVG(marks) AS avg_marks
FROM student
GROUP BY dept_name
HAVING AVG(marks) > 75;
```

2. Departments having more than 1 student

```
SELECT dept_name, COUNT(*) AS total_students
FROM student
GROUP BY dept_name
HAVING COUNT(*) > 1;
```

3. Departments where total fees > 65,000

```
SELECT dept_name, SUM(fees) AS total_fees
FROM student
GROUP BY dept_name
HAVING SUM(fees) > 65000;
```

🔷 3️⃣ Real-Time Scenarios

🎓 Example 1 – College Database

You want to find each department's:

Average marks

Total fees collected

Only those departments where average marks > 75

This is a GROUP BY + HAVING use case.

💼 Example 2 – Company Project (Sales Report)

Table: sales_data(region, product, sales_amount)

Requirement:

Show total sales per region but only for regions where total sales > ₹10,00,000.

```
SELECT region, SUM(sales_amount) AS total_sales
FROM sales_data
GROUP BY region
HAVING SUM(sales_amount) > 1000000;
```

Used in:

Sales dashboards, management reports, and performance summaries.

◈ 4️⃣ Real-Time Project Usage

Domain	Example Use Case	SQL Logic
🎓 College ERP	Find average marks per department	GROUP BY dept_name
💰 Banking System	Find total deposits per branch	GROUP BY branch_id
🛒 E-Commerce	Find total sales per category	GROUP BY category_id
🏢 HR/Payroll	Find average salary per department	GROUP BY dept_id
📊 Finance	Find customers spending > ₹50,000	HAVING SUM(amount) > 50000

◈ 5️⃣ Interview Questions (Beginner → Expert)

Level	Question	Example Answer
🌼 Beginner	What is the use of GROUP BY?	It groups rows having the same values and applies aggregation.
🌼 Beginner	Difference between WHERE and HAVING?	WHERE filters rows before grouping; HAVING filters groups after grouping.

Level	Question	Example Answer
⚙ Intermediate	Can we use WHERE and HAVING in the same query?	Yes. WHERE filters rows, HAVING filters grouped results.
⚙ Intermediate	Find departments where total fees > ₹60,000.	Use SUM(fees) with HAVING.
🚀 Expert	Can we use alias in HAVING?	In PostgreSQL, yes. Example: HAVING avg_marks > 80.
🚀 Expert	How does GROUP BY work internally?	DB sorts rows by grouping columns, then applies aggregation per group.

🔷 6. Real-Time Practice Questions

1. Show department-wise average marks but only those > 75.
2. Find number of students per department where count > 2.
3. Show department-wise total fees where total fees > 60,000.
4. Find departments with maximum marks > 85.
5. In a sales table, show products whose average monthly sales > ₹50,000.

🔷 7. Key Differences: WHERE vs GROUP BY vs HAVING vs ORDER BY

Clause	Purpose	Execution Order
WHERE	Filters individual rows	Before grouping
GROUP BY	Groups rows	Before HAVING
HAVING	Filters grouped results	After grouping
ORDER BY	Sorts the final result	At the end

◈ 8️⃣ Real-Time Explanation (Simple Classroom Language)

"First, we use WHERE to filter individual rows.

Then we use GROUP BY to group similar data (like department-wise).

Finally, we use HAVING to filter the grouped results (like show only groups where average marks > 75)."

This is how reports are generated in real-world projects such as:

Student Reports

Sales Dashboards

Department Statistics

Branch-wise Performance

◈ 9️⃣ Quick Summary

Concept	**Description**
GROUP BY	Groups data based on column values.
HAVING	Filters groups using aggregate functions.
WHERE	Filters rows before grouping.
ORDER BY	Sorts the final output.
Real-time Use	Used in reports, dashboards, and analytics queries.

✅ In Short:

Use GROUP BY to group rows.

Use HAVING to filter groups.

Always use GROUP BY before HAVING.

Use ORDER BY at the end for sorting.

💮 GROUP BY మరియు HAVING అంటే ఏమిటి?

👉 GROUP BY

డేటాను ఒక "సమూహం" (group) ప్రకారం విభజించి, దానిపై aggregate functions (SUM, AVG, COUNT, MAX, MIN...) చేయడానికి ఉపయోగిస్తాం.

👉 HAVING

GROUP BY చేసిన తరాత్ర వచిన్న groups పై filter condition పెట్టడానికి వాడతాం.

(WHERE వాడలేము ఎందుకంటే WHERE individual rows పై పనిచేస్తుంది, HAVING groups పై పనిచేస్తుంది.)

🤔 GROUP BY ఉపయోగం

📦 Syntax:

SELECT column_name, AGGREGATE_FUNCTION(column_name)

FROM table_name

GROUP BY column_name;

🎓 Example 1: Department-wise total salary

SELECT dept_id, SUM(salary) AS total_salary

FROM employee

GROUP BY dept_id;

🔚 Output: ప్రతి department కి మొత్తం salary ఎంత ఉందో చూపిస్తుంది.

dept_id total_salary

1 55000

2 133000

3 48000

NULL 53000

🎓 Example 2: Department-wise average salary

SELECT dept_id, AVG(salary) AS avg_salary

FROM employee

GROUP BY dept_id;

➡ ప్రతి department లో average salary చూపిస్తుంది.

🎓 Example 3: HAVING Clause

Requirement:

“Average salary 50,000 కన్నాఎక్కువ ఉన్నdepartments మాత్రమే చూపించాలి”

SELECT dept_id, AVG(salary) AS avg_salary

FROM employee

GROUP BY dept_id

HAVING AVG(salary) > 50000;

➡ HAVING ఉపయోగించడం వల్ల filter condition groups మీద పడుతుంది.

⚖️ WHERE vs HAVING తేడా

Concept	**ఎప్పుడు వాడాలి (When to Use)**	**ఎక్కడ పనిచేస్తుంది (Where It Works)**
WHERE	Group చేయడానికి ముందు	Individual rows
HAVING	Group చేసిన తరువాత	Group results

ఉదా:

WHERE before grouping

```
SELECT dept_id, AVG(salary)
FROM employee
WHERE salary > 40000
GROUP BY dept_id;
```

HAVING after grouping

```
SELECT dept_id, AVG(salary)
FROM employee
GROUP BY dept_id
HAVING AVG(salary) > 50000;
```

🎯 రియల్ టైమ్ ఉదాహరణలు

Requirement	**SQL Concept**	**Example**
ప్రతి department లో total employees	GROUP BY + COUNT	SELECT dept_id, COUNT(*) FROM employee GROUP BY dept_id;
Average salary > 60000 ఉన్నdepartments	GROUP BY + HAVING	HAVING AVG(salary) > 60000;
HR కి total salary report కావాలి	GROUP BY + SUM	SUM(salary)

💠 GROUP BY & HAVING – Full Practical Concept (PostgreSQL)

📦 1️⃣ Table Creation (Sample Data)

```
DROP TABLE IF EXISTS sales;

CREATE TABLE sales (
    sale_id SERIAL PRIMARY KEY,
    salesman_name VARCHAR(50),
    region VARCHAR(50),
    sale_amount NUMERIC(10,2),
    sale_date DATE
);

INSERT INTO sales (salesman_name, region, sale_amount, sale_date)
VALUES
('Ravi', 'South', 5000, '2025-10-01'),
('Ravi', 'South', 7000, '2025-10-03'),
('Sneha', 'North', 8000, '2025-10-01'),
('Arjun', 'East', 6000, '2025-10-04'),
('Arjun', 'East', 3000, '2025-10-05'),
('Deepa', 'West', 4000, '2025-10-01'),
('Deepa', 'West', 7000, '2025-10-02'),
('Kiran', 'South', 2000, '2025-10-06');
```

🎯 2️⃣ Basic GROUP BY Example

Requirement:

ప్రతి salesman ఎంత మొత్తం sales చేశాడు?

```
SELECT salesman_name, SUM(sale_amount) AS total_sales
FROM sales
GROUP BY salesman_name;
```

Result:

salesman_name	total_sales
Ravi	12000
Sneha	8000
Arjun	9000
Deepa	11000
Kiran	2000

🎯 3️⃣ GROUP BY Multiple Columns

Requirement:

ప్రతి salesman ప్రతి region లో ఎంత sales చేశాడు?

```
SELECT salesman_name, region, SUM(sale_amount) AS total_sales
FROM sales
GROUP BY salesman_name, region;
```

Result:

salesman_name	region	total_sales
Ravi	South	12000
Sneha	North	8000
Arjun	East	9000
Deepa	West	11000
Kiran	South	2000

🎯 4️⃣ HAVING Clause Example

Requirement:

మొత్తం sales ₹10,000 కన్నా ఎక్కువ ఉన్న salesmen మాత్రమే చూపించాలి.

```
SELECT salesman_name, SUM(sale_amount) AS total_sales
FROM sales
GROUP BY salesman_name
HAVING SUM(sale_amount) > 10000;
```

Result:

salesman_name	total_sales
Ravi	12000
Deepa	11000

🎯 5️⃣ WHERE + GROUP BY + HAVING Together

Requirement:

October 2 తరాత్ చేసిన sales consider చేయాలి

మరియు total sales ₹7,000 కన్నా ఎక్కువ ఉన్నవారిని మాత్రమే చూపించాలి.

```
SELECT salesman_name, SUM(sale_amount) AS total_sales
FROM sales
WHERE sale_date >= '2025-10-02'
GROUP BY salesman_name
HAVING SUM(sale_amount) > 7000;
```

🧠 WHERE vs HAVING Visualization

[Rows in Table]

↓

WHERE Filter

↓

GROUP BY Groups

↓

HAVING Filter on Groups

↓

Final Result

👉 WHERE – individual rows filter చేస్తుంది

👉 HAVING – grouped results filter చేస్తుంది

💮 GROUP Formation Diagram (Text Visualization)

ఇకపై sales table data ని region-wise group చేసినట్లయితే ఇలా కనిపిస్తుంది 👇

Original Data:

Ravi | South | 5000

Ravi | South | 7000

Sneha | North | 8000

Arjun | East | 6000

Arjun | East | 3000

Deepa | West | 4000

Deepa | West | 7000

Kiran | South | 2000

After GROUP BY region:

South -> (Ravi-5000, Ravi-7000, Kiran-2000)

North -> (Sneha-8000)

East -> (Arjun-6000, Arjun-3000)

West -> (Deepa-4000, Deepa-7000)

Now apply SUM():

Region	Total Sales
South	14000
North	8000
East	9000
West	11000

Now apply HAVING (Total > 10000):

Region	Total Sales
South	14000
West	11000

🏢 Real-time Scenarios

Real-time Requirement Example Query

HR report: ప్రతి department లో employees count SELECT dept_id, COUNT(*) FROM employee GROUP BY dept_id;

Sales report: ప్రతి salesman కి total sales SELECT salesman_name, SUM(sale_amount) FROM sales GROUP BY salesman_name;

Finance report: ₹10,000 కంటే ఎక్కువ revenue ఉన్నregions HAVING SUM(sale_amount) > 10000;

Education: ప్రతి department లో average marks SELECT dept_id, AVG(marks) FROM students GROUP BY dept_id;

==========
==========
==========
==========

Student example

- student + department tables
- realistic sample data
- GROUP BY, HAVING, WHERE practical queries
- step-by-step explanation + diagrams

1. Table Creation

```
DROP TABLE IF EXISTS department;
DROP TABLE IF EXISTS student;

CREATE TABLE department (
    dept_id SERIAL PRIMARY KEY,
    dept_name VARCHAR(50)
);
```

```
CREATE TABLE student (
    student_id SERIAL PRIMARY KEY,
    student_name VARCHAR(50),
    dept_id INT REFERENCES department(dept_id),
    marks INT,
    grade CHAR(1)
);
```

📝 2️⃣ Insert Sample Data

```
INSERT INTO department (dept_name) VALUES
('CSE'),
('ECE'),
('EEE'),
('MECH');

INSERT INTO student (student_name, dept_id, marks, grade) VALUES
('Ravi', 1, 85, 'A'),
('Sneha', 1, 90, 'A'),
('Arjun', 2, 78, 'B'),
('Deepa', 2, 88, 'A'),
('Kiran', 3, 60, 'C'),
('Lalitha', 3, 72, 'B'),
('Manoj', 4, 55, 'D'),
('Sujatha', 4, 65, 'C'),
('Ramesh', 1, 95, 'A'),
('Pooja', 2, 82, 'B');
```

🎯 3️⃣ Basic GROUP BY Example

Requirement:

ప్రతి department లో average marks ఎంత?

```
SELECT d.dept_name, AVG(s.marks) AS avg_marks
FROM student s
JOIN department d ON s.dept_id = d.dept_id
GROUP BY d.dept_name;
```

Result:

dept_name avg_marks

CSE 90.00

ECE 82.67

EEE 66.00

MECH 60.00

🎯 4️⃣ GROUP BY + HAVING Example

Requirement:

Average marks 70 కంటే ఎక్కువ ఉన్న departments మాత్రమే చూపించాలి.

```
SELECT d.dept_name, AVG(s.marks) AS avg_marks
FROM student s
JOIN department d ON s.dept_id = d.dept_id
GROUP BY d.dept_name
HAVING AVG(s.marks) > 70;
```

Result:

dept_name avg_marks

CSE 90.00

ECE 82.67

🎯 5️⃣ WHERE + GROUP BY + HAVING Example

Requirement:

Grade ‘A’ ఉన్న students మాత్రమే consider చేయాలి,

మరియు average marks 80 కంటే ఎక్కువ ఉన్న departments మాత్రమే చూపించాలి.

```
SELECT d.dept_name, AVG(s.marks) AS avg_marks
FROM student s
JOIN department d ON s.dept_id = d.dept_id
WHERE s.grade = 'A'
GROUP BY d.dept_name
HAVING AVG(s.marks) > 80;
```

Result:

dept_name avg_marks

CSE 90.00

ECE 88.00

🎓 6️⃣ Diagram – GROUP BY Visual Representation

Student Table (Marks)

Ravi | CSE | 85

Sneha | CSE | 90

Ramesh | CSE | 95

Arjun | ECE | 78

Deepa | ECE | 88

Pooja | ECE | 82

Kiran | EEE | 60

Lalitha | EEE | 72

Manoj | MECH | 55

Sujatha | MECH | 65

After GROUP BY dept_name:

CSE -> (85, 90, 95) → AVG = 90

ECE -> (78, 88, 82) → AVG = 82.67

EEE -> (60, 72) → AVG = 66

MECH -> (55, 65) → AVG = 60

After HAVING (AVG > 70):

CSE | 90

ECE | 82.67

🏫 Real-time College Scenarios

Requirement	**SQL Concept**	**Example**
ప్రతి department లో average marks తెలుసుకోవాలి	GROUP BY	AVG(marks)
70 కన్నాఎకుఫ average ఉన్న departments	GROUP BY + HAVING	HAVING AVG(marks) > 70
Grade 'A' ఉన్నారి మాత్రమే తీసుకోవాలి	WHERE + GROUP BY + HAVING	WHERE grade='A' HAVING AVG(marks) > 80
ప్రతి department లో టాప్ performance report	GROUP BY + MAX	SELECT dept_name, MAX(marks)

🧠 Concept Summary

Keyword	**Use**	**Example**
GROUP BY	Group చేయడానికి	Dept-wise, Grade-wise reports
HAVING	Group filter చేయడానికి	Top departments only
WHERE	Rows filter చేయడానికి	Grade = 'A', Dept = 'CSE' వంటి filters