Video-11

Topics to cover:

- Group By Clause
- Having Clause
- Difference Between Where and Having Clause
- Group_Concat

Group By

GROUP BY is used to arrange rows into groups based on the values of one or more columns.

- It is usually combined with aggregate functions (COUNT, SUM, AVG, MAX, MIN) to get summary results for each group.

Syntax:

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

Group By

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

```
SELECT country, COUNT(*) AS number
FROM Customers
GROUP BY country;
```

country	number
UAE	1
UK	2
USA	2

Creating orders table to understand Group By

```
CREATE TABLE orders (
    cust_id   INT,
    order_id  INT,
    country   VARCHAR(50),
    state     VARCHAR(50)
);

INSERT INTO orders (cust_id, order_id, country, state) VALUES
(1, 100, 'USA', 'Seattle'),
(2, 101, 'INDIA', 'Delhi'),
(2, 103, 'INDIA', 'Delhi'),
(4, 108, 'USA', 'WDC'),
(5, 109, 'UK', 'London'),
(4, 110, 'USA', 'WDC'),
(3, 120, 'INDIA', 'Delhi'),
(2, 121, 'INDIA', 'Goa'),
(1, 131, 'USA', 'Seattle'),
(6, 142, 'USA', 'Seattle'),
(7, 150, 'USA', 'Seattle');

SELECT * FROM orders;
```

-- Example-1: Calculate total order placed Country wise.

```
SELECT country, count(order_id) as order_count
FROM orders
GROUP BY country;

-- You can also use * in place of column name.

SELECT country, count(*) as order_count
FROM orders
GROUP BY country;
```

-- Example-2: Calculate total order placed State wise.

```
SELECT state, count(order_id) as order_count
FROM orders
GROUP BY state;
```

-- Example-3: Calculate total order placed by both Country & State.

```
SELECT country, state, count(order_id) as order_count
FROM orders
GROUP BY country, state;
```

Creating employees table to understand Group By

```
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    department VARCHAR(50),
    salary DECIMAL(10,2),
    hire_date DATE );

INSERT INTO employees (emp_id, first_name, last_name, department, salary, hire_date)
VALUES
(1, 'Amit', 'Sharma', 'IT', 60000, '2020-01-15'),
(2, 'Neha', 'Rao', 'IT', 75000, '2021-03-10'),
(3, 'Vikram', 'Joshi', 'IT', 70000, '2019-07-25'),
(4, 'Anita', 'Deshmukh', 'HR', 80000, '2022-05-05'),
(5, 'Rohan', 'Iyer', 'HR', 50000, '2018-11-20'),
(6, 'Karan', 'Patel', 'HR', 55000, '2021-02-18'),
(7, 'Meera', 'Kapoor', 'Sales', 45000, '2020-09-12'),
(8, 'Sanjay', 'Verma', 'Sales', 48000, '2019-11-30'),
(9, 'Priya', 'Nair', 'Marketing', 65000, '2021-07-14'),
(10, 'Arjun', 'Singh', 'Marketing', 62000, '2022-03-08');

Select * from employees;
```

```
-- Example-4: Calculate Total salary per department
SELECT department, sum(salary)
FROM employees
GROUP BY department;

-- Example-5: Calculate Average salary per department
SELECT department, avg(salary)
FROM employees
GROUP BY department;

-- Example-6: Group by with multiple aggregation
SELECT department,
       COUNT(*) AS total_employees,
       AVG(salary) AS avg_salary,
       MIN(salary) AS lowest_salary,
       MAX(salary) AS highest_salary
FROM employees
GROUP BY department;
```

Having

The HAVING clause in SQL is used to filter grouped records.

The HAVING clause in SQL is used to filter groups of rows after applying GROUP BY and is especially useful when applying conditions to aggregate functions like SUM(), COUNT(), AVG(), etc.

Where Clause V/s Having Clause

- WHERE filters rows before grouping.
- HAVING filters groups after grouping.
- HAVING Typically used with GROUP BY, but can also be used without it (then it acts like WHERE).

Syntax:

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

Understanding Difference Between Where and Having

```
Select * from employees;

Select *
from employees
Where department <> 'IT';

Select department, sum(salary)
from employees
group by department;

Select department, sum(salary)
from employees
group by department
having sum(salary) > 100000;
```

GROUP_CONCAT

- GROUP_CONCAT() function returns a string with a concatenated non-NULL value from a group.
- It is used to concatenate values from multiple rows within a specific column into a single string.
- It is particularly useful for combining and displaying related data in a compact format.

Syntax:

GROUP_CONCAT([DISTINCT] expression

[ORDER BY expression ASC|DESC]

[SEPARATOR string])

Group_Concat Examples:

```
-- Concat state for each country
SELECT country, GROUP_CONCAT(state) AS state_in_country
FROM orders
GROUP BY country;

-- Concat distinct state for each country
SELECT country, GROUP_CONCAT(DISTINCT state) AS state_in_country
FROM orders
GROUP BY country;

-- Concat distinct state for each country and sort it in descending order
SELECT country, GROUP_CONCAT(DISTINCT state ORDER BY STATE DESC) AS state_in_country
FROM orders
GROUP BY country;

-- Concat distinct state for each country and sort it in ascending order and use "|" as separator
SELECT country, GROUP_CONCAT(DISTINCT state ORDER BY STATE SEPARATOR "|") AS
state_in_country
FROM orders
GROUP BY country;
```