----------LECTURE- 6----------

1 WHERE Clause

Used to filter rows based on a condition.

☆ **Syntax**

```
SELECT columns
FROM table_name
WHERE condition;
```

Example

```
SELECT * FROM employee WHERE city = 'Delhi';
```

---

2 ORDER BY Clause

Used to sort data.

☆ **Syntax**

```
ORDER BY column_name ASC;   -- ascending (default)
ORDER BY column_name DESC;  -- descending
```

Example

```
SELECT * FROM employee ORDER BY salary DESC;
```

---

3 GROUP BY Clause

Used to group rows based on a column → used with aggregate functions.

☆ **Syntax**

```
SELECT column, SUM(salary)
FROM employee
GROUP BY column;
```

Example

```
SELECT department, AVG(salary)
FROM employee
GROUP BY department;
```

---

4 HAVING Clause

Same as WHERE but used after grouping.

☆ Syntax

```
SELECT department, AVG(salary)
FROM employee
GROUP BY department
HAVING AVG(salary) > 30000;
```

☆ WHERE vs HAVING

WHERE	HAVING
Filters rows before grouping	Filters groups after grouping
Cannot use aggregate functions	Can use aggregate functions

Example:

✕ Wrong:

```
WHERE SUM(salary) > 30000;
```

✓ Correct:

```
HAVING SUM(salary) > 30000;
```

---

5 LIMIT and OFFSET

LIMIT → restrict number of rows

```
SELECT * FROM employee LIMIT 5;
```

OFFSET → skip rows (must use with LIMIT)

```
SELECT * FROM employee LIMIT 1 OFFSET 2;
```

Meaning:

Skip first 2 rows → show 1 row.

☆ **Note**
✓ **OFFSET works only with LIMIT**
✓ **LIMIT can work without OFFSET**

---

6 DISTINCT

Used to remove duplicate values.

Example

```
SELECT DISTINCT city FROM employee;
```

---

7 BETWEEN

Used to select values between a range (inclusive).

Example

```
SELECT * FROM employee
WHERE salary BETWEEN 20000 AND 40000;
```

---

8 IN and NOT IN

Example

```
SELECT * FROM employee
WHERE city IN ('Delhi', 'Mumbai', 'Noida');

SELECT * FROM employee
WHERE department NOT IN ('HR', 'Sales');
```

---

9 LIKE (pattern matching)

% → any number of characters
_ → exactly one character

Examples:

```
SELECT * FROM users WHERE name LIKE 'A%';   -- starts with A
SELECT * FROM users WHERE name LIKE '%a';   -- ends with a
SELECT * FROM users WHERE name LIKE '_a%';  -- second letter is a
```

---

10 IS NULL / IS NOT NULL

```
SELECT * FROM employee WHERE email IS NULL;
SELECT * FROM employee WHERE phone IS NOT NULL;
```

---

11 AND / OR Operators

AND → both conditions must be true

```
WHERE salary > 30000 AND city = 'Delhi';
```

OR → either condition true

```
WHERE city='Delhi' OR city='Mumbai';
```

---

12 Aggregate functions

Used to perform calculations on multiple rows.

Function	What it does
SUM()	Adds values
MAX()	Highest value
MIN()	Lowest value
COUNT()	Number of rows
AVG()	Average value

Example

```
SELECT SUM(salary) FROM employee;
```

---

13 COUNT(*) vs COUNT(column_name)

✓ COUNT(*)
Counts all rows (including rows with NULL values).

✓ COUNT(column_name)
Counts only non-NULL values.

Example:

If table has:

id	email
1	a@g.com
2	NULL
3	b@g.com

COUNT(*) → 3
COUNT(email) → 2

---

14 Selecting only NULL values

```
SELECT COUNT(*) FROM employee
WHERE email IS NULL;
```

---

15 COUNT(NULL)

Always returns 0

Because COUNT(column) ignores NULL.

```
SELECT COUNT(NULL);   -- Output: 0
```

---

16 Count only NULL using IF()

Example

```
SELECT COUNT(IF(email IS NULL, 1, 0))
FROM employee;
```

Explanation:

- **If email is NULL → 1**
- **If email is NOT NULL → 0**
  COUNT counts only 1's = how many NULLs

---

17 Difference between != , <> and NOT

All mean NOT EQUAL.

Operator	Meaning	
!=	Not equal	
<>	Not equal	
NOT	Negation	

Examples:

```
WHERE salary != 30000;
WHERE salary <> 30000;
WHERE NOT salary = 30000;
```

All give same output.

---

18 Finding 2nd Highest Salary (Using OFFSET)

Example:

```
SELECT * FROM employee
ORDER BY salary DESC
LIMIT 1 OFFSET 1;
```

Breakdown:

- Order by salary descending
- Skip 1 row (highest salary)
- Show next row → 2nd highest

---

19 Example

☆ Select DISTINCT

```
SELECT DISTINCT city FROM employee;
```

☆ Select COUNT(NULL)

```
SELECT COUNT(NULL);   -- Always 0
```

☆ Select ONLY NULL rows

```
SELECT COUNT(*) FROM employee WHERE email IS NULL;
```

☆ IF condition example

```
SELECT COUNT(IF(email IS NULL, 1, 0)) FROM employee;
```