# ⍰ PHASE-4 SQL NOTES (Easy + Crystal Clear)

## 📌 Topics Covered

✔ LIKE
✔ NOT LIKE
✔ Wildcards (% _ )
✔ IN / NOT IN (Advanced logic)
✔ NULL Handling (IS NULL / IS NOT NULL)
✔ ORDER BY (Multi-column sorting)
✔ Practical business filters

## 🔥 1. LIKE Operator

Used to search text patterns.

### ⍰ Syntax

WHERE column LIKE 'pattern'

### 📌 Wildcards with LIKE

| Wildcard | Meaning | Example |
|---|---|---|
| % | Any number of characters | 'A%' → Starts with A |
| _ | Only 1 character | '_a%' → second letter is "a" |

### 📕 Examples

1. **Starts with A**

WHERE name LIKE 'A%'

2. **Ends with n**

WHERE name LIKE '%n'

3. **Contains "ai"**

WHERE name LIKE '%ai%'

4. **Second letter is 'a'**

WHERE name LIKE '_a%'

---

## 🔥 2. NOT LIKE

Used to **exclude** text patterns.

### Examples

1. Not starting with A

WHERE name NOT LIKE 'A%'

2. Not containing 'in'

WHERE name NOT LIKE '%in%'

## 🔥 3. Wildcards (Deep Understanding)

### % → Many characters

| Pattern | Meaning |
|---|---|
| 'A%' | A… anything |
| '%A' | …A |
| '%A%' | contains A |

### _ → one character

| Pattern | Meaning |
|---|---|
| '_a%' | second letter is a |
| 'J___' | J + 3 letters |

### ⍰ Does LIKE work on numbers?

✔ **YES — *but only if the column is stored as text (VARCHAR / CHAR)***

⍰ **NO — *if the column is stored as a numeric type (INT, BIGINT, FLOAT, DECIMAL) Like : on salary column.***

**This is the correct rule.**

---

### 📌 Why LIKE does NOT work on numeric columns?

Because:

- LIKE is a **string pattern matching operator**
- Numbers cannot accept wildcards (%, _)
- So SQL gives an error or converts internally (not recommended)

Example (WRONG on INT column):

WHERE salary LIKE '500%'
***This will NOT work if salary is INT.**

---

### 📌 BUT % wildcard always works WITH LIKE only

% is **NOT** a numeric operator.
You cannot use % without LIKE.
Example (this is wrong):
WHERE salary % 500
The % in SQL is ONLY for LIKE pattern matching.

| Pattern | Meaning |
|---|---|

### 🎯 Correct Understanding (Very Important)

✔ **LIKE works only on text**
✔ **% and _ are string wildcards**
✔ **If you try LIKE on numeric:**

- SQL Server → Converts number → Still not recommended
- MySQL → Converts → Works but slow
- PostgreSQL → Throws error

So ALWAYS use LIKE only on text columns.

### 🚀 Example to make it 100% clear

⍰ **Salary column = INT**
This will NOT work:
WHERE salary LIKE '50%'
✔ **Salary converted to VARCHAR**
This will work:
WHERE CAST(salary AS VARCHAR) LIKE '50%'

---

## Below is a **clean theory + logic explanation** covering:

✔ Advanced wildcards
✔ Range matches
✔ Multiple conditions
✔ Escaped patterns
✔ LIKE + NOT LIKE logic
✔ Text vs numeric casting
✔ Important functions (LEFT, RIGHT, LEN, DATE functions, CAST)
✔ Interview-style understanding

This is everything you need to **master wildcard + functions** the way interviewers expect.

# 🔥 1. ADVANCED WILDCARDS (LIKE PATTERNS)

### [] → Match ANY character in the list (SQL Server only)

'[abc]%' → starts with a OR b OR c
'[0-9]%' → starts with any digit

### [^] → NOT these characters

'[^a-z]%' → does NOT start with lowercase letters

**[abc] and %abc% look similar but their meaning is COMPLETELY different.**

Here is the clearest explanation 👇

| Pattern | Meaning |
|---|---|

## ✔ 1. [abc] — Character CLASS (matches ONLY ONE character)

### Meaning:

[abc] means the **first character** (or the position where you use it) can be:

- **a**
- **b**
- **c**

Only **one** of them.

### Example:

```
WHERE name LIKE '[abc]%'
```

This means:

✔ Name starts with **a** OR **b** OR **c**

Matches:

- ankit
- batman
- cat
- cable

Does NOT match:

- david
- ram
- 1start

**KEY POINT:**

➡ [abc] NEVER looks for "abc together"

➡ It only checks **one character** from the set.

```
WHERE name LIKE '%[abc]%'
```

Your words:

paresh → contains a ✔
biswa → contains b ✔
cat → contains c and a ✔
happy → contains a ✔
All are perfect matches. ✔✔✔
Because %[abc]% means:
☞ The text contains any one letter from the set a, b, or c
(at any position)

## ✔ [ ^ ] in SQL LIKE — "NOT these characters"

Inside square brackets [ ], the **circumflex** ^ means:

| Pattern | Meaning |
|---|---|

**▯ Do NOT allow any of these characters.**

It is used to EXCLUDE characters.

---

## ✔ Syntax:

```
WHERE name LIKE '[^abc]%'
```

### Meaning:

☞ The **first character** must be **NOT a**, **NOT b**, and **NOT c**
(should be anything except a, b, or c)

---

## 🔥 Examples

### Pattern:

```
LIKE '[^abc]%'
```

### Matches (Correct ✔)

- **david** (starts with d → allowed)
- **ram** (starts with r → allowed)
- **happy** (starts with h → allowed)
- **zebra** (starts with z → allowed)

### Does NOT match (▯)

- **apple** (starts with **a**)
- **ball** (starts with **b**)
- **cat** (starts with **c**)

Because these start with **a / b / c**, which are NOT allowed.

---

## ✔ Example 2 — exclude digits

```
WHERE name LIKE '[^0-9]%'
```

Meaning:
☞ first character should NOT be any digit

| Pattern | Meaning |
|---|---|

Matches:

- paresh
- biswa
- hello

Does NOT match:

- 8ball
- 2cool

---

### ✔ Example 3 — exclude vowels

```
WHERE word LIKE '[^aeiou]%'
```

Matches:

- ball
- cricket
- table

Does NOT match:

- apple
- elephant
- orange

---

### ✔ 2. `%abc%` — Substring match (matches full sequence “abc”)

**Meaning:**

`%abc%` means:

✔ The text contains **abc exactly in this order**

Example:

```
WHERE name LIKE '%abc%'
```

Matches:

- xabcx
- 123abc
- aabc
- abcdef

Does NOT match:

- acb

| Pattern | Meaning |
|---|---|

- bac
- abdc

**KEY POINT:**

➡ %abc% matches **the full 3-character sequence**

➡ They must appear **together** as a → b → c.

## 🔥 2. RANGE MATCHING

Ranges are allowed inside brackets **[ ]**

Examples:

```
LIKE '[A-F]%'    → A to F
LIKE '[0-5]%'    → digits 0 to 5
LIKE '%[A-Z]'    → ends with capital A–Z
```

## 🔥 3. MULTIPLE CONDITIONS (LIKE + AND/OR/NOT)

### Combine LIKE patterns

```
WHERE name LIKE 'A%'
  OR name LIKE 'B%'
```

### Exclude patterns

```
WHERE city NOT LIKE '%Town%'
```

### LIKE + NOT LIKE together

```
WHERE name LIKE 'A%'
 AND name NOT LIKE '%test%'
```

This is used heavily in interviews → pattern filtering.

## 🔥 4. ESCAPED PATTERNS

If you want to search for the literal characters:

- %
- _
- [

You must escape them.

Example (SQL Server):

```
WHERE code LIKE '%\%%' ESCAPE '\'
```

Meaning: find text **that contains % symbol**.

## ✔ Why ESCAPE '\' is needed?

Because SQL must know:

- \ is the escape character
- \% means literal %
- % without escape means wildcard

So the syntax is correct only when you add:

```
ESCAPE '\'
```

| Pattern | Meaning |
|---|---|

## 🔥 5. TEXT vs NUMERIC CASTING IN LIKE

**LIKE works only on text, even if the column is numeric.**

If the column is numeric:

```
WHERE CAST(salary AS VARCHAR) LIKE '5%'
```

Example: find all salaries starting with 5 (50,000 / 5000 / 5 lakh)

**Opposite: numeric comparison**

```
WHERE salary > 50000
```

## 🔥 6. IMPORTANT STRING FUNCTIONS (Used with LIKE Frequently)

### 1️⃣ LEFT (column, length)

Returns the first N characters.

Example:

```
WHERE LEFT(name, 1) = 'A'
```

Same as:

```
WHERE name LIKE 'A%'
```

### 2️⃣ RIGHT (column, length)

Returns last N characters.

Example:

```
WHERE RIGHT(phone, 2) = '91'
```

Same as:

```
WHERE phone LIKE '%91'
```

### 3️⃣ LEN (column)

Length of the text.

Example:

```
WHERE LEN(name) = 5
```

Find all names of exactly 5 characters.

### 4️⃣ SUBSTRING (column, start, length)

Extract a part of the string.

```
SELECT SUBSTRING(name, 2, 3)
```

## 🔥 7. DATE FUNCTIONS Used With LIKE

**Convert date to text → pattern match**

```
WHERE CONVERT(VARCHAR(10), order_date, 120) LIKE '2024-05%'
```

Find all orders from May 2024.

**DATEPART**

```
WHERE DATEPART(month, order_date) = 5
```

**YEAR / MONTH / DAY**

| Pattern | Meaning |
|---|---|
| WHERE YEAR(order_date) = 2024<br>WHERE MONTH(order_date) = 12 | |

## 🔥 8. CAST / CONVERT (Very Common in Interviews)

### Convert number to text → LIKE:

```
WHERE CAST(emp_id AS VARCHAR) LIKE '10%'
```

### Convert string to number:

```
WHERE CAST(age AS INT) > 25
```

### Convert date to text:

```
SELECT CONVERT(VARCHAR, hire_date, 103)
```

## 🔥 9. HIGH-LEVEL INTERVIEW PATTERNS

### 1️⃣ Find emails from Gmail only

```
WHERE email LIKE '%@gmail.com'
```

### 2️⃣ Find Indian phone numbers starting with 9, 8, or 7

```
WHERE phone LIKE '[987]%'
```

### 3️⃣ Validate PAN number pattern

```
WHERE pan LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9][A-Z]'
```

### 4️⃣ Find employees with 5-letter names

```
WHERE LEN(name) = 5
```

### 5️⃣ Remove special characters using ESCAPE

```
WHERE product_code LIKE '%\_%' ESCAPE '\'
```

## 🔥 10. COMPLETE INTERVIEW SUMMARY

### Most important things to remember:

✔ % → many characters
✔ _ → one character
✔ [] → ranges
✔ [^] → NOT ranges
✔ Escape literal % _ [ using ESCAPE
✔ LIKE works on **text only**, not numbers
✔ CAST/CONVERT helps convert
✔ LEFT / RIGHT / LEN frequently used with patterns
✔ Date → convert to string for pattern matching

- **Because most DE SQL questions related to LIKE will fall under:**

- Starts with
- Ends with
- Contains
- Second/third letter matching
- Simple pattern search
- Email/domain filtering
- File name filtering
- Prefix/suffix based logic

You already understand **ALL** of these.

## 🔵 4. IN & NOT IN (Advanced Logic)

### Simple IN

```
WHERE city IN ('Delhi', 'Chennai', 'Bangalore')
```

### Simple NOT IN

```
WHERE city NOT IN ('Delhi', 'Chennai')
```

### 🔥 Deep Logic Notes

| Condition | Meaning |
|---|---|
| salary NOT IN (50000) | salary ≠ 50000 |
| age NOT IN (25, 30) | age ≠ 25 and age ≠ 30 |
| city NOT IN ('Delhi', 'Mumbai') | exclude both |

---

## 🔥 5. NULL Handling

NULL = blank / no value.

### Check NULL

```
WHERE email IS NULL
```

### Check NOT NULL

```
WHERE email IS NOT NULL
```

⍰ Do NOT use = NULL or != NULL ⍰
They never work.

---

## 🔥 6. ORDER BY with Multiple Columns

Sorting by more than one column.

```
ORDER BY department ASC, salary DESC
```

Meaning:
1️⃣ Sort by department A → Z
2️⃣ Inside each department, show salary high → low

---

## 🔥 7. Practical Filters (Important)

### Age between a range

```
WHERE age BETWEEN 25 AND 35
```

### Excluding multiple departments

```
WHERE department NOT IN ('HR', 'Finance')
```

### Checking emails with Gmail

```
WHERE email LIKE '%gmail.com'
```

## 🚀 Conclusion: Are you ready?

**✔ Your wildcard knowledge = More than enough to clear interviews**

**✔ You already understand the 2 wildcards fully:**

- **%** = multi-character
- **_** = single character

**💯 You are now ABOVE AVERAGE in SQL fundamentals.**