ScopeTreasure Courses

SQL COURSES · LESSON DB.4 · CODE SCOPE SERIES

WHERE, ORDER BY, LIMIT — Chhanna Aur Tartib

Filter and arrange your answers: WHERE with AND/OR, LIKE, IN and BETWEEN, sorting with ORDER BY, and top results with LIMIT.

Introduction

`SELECT * FROM students;` answers "show everything". Real questions are sharper: *students of class 10-A who scored above 85*. *The top 3 students of the school. Everyone from Lahore or Multan.* This lesson sharpens the two halves of every real question: **WHERE** filters *which rows* come back (chhanna — sieving), and **ORDER BY + LIMIT** arrange *in what order, and how many* (tartib — ordering).

Our five-student table is the test data throughout — small enough that you can predict every answer before reading it. Predicting first, then checking, is how SQL really gets learned.

(Roman Urdu: WHERE = chhalni, jo rows chahiye wohi aayengi; ORDER BY = tartib, LIMIT = bas itne hi.)

Key points

- **WHERE compares every row:** the condition is tested row by row; only rows where it is **true** come back. Comparison signs: `=`, `!=` (not equal), `>`, `<`, `>=`, `<=`.
- **AND needs both true; OR needs either true:** `class = '10-A' AND marks > 85` is strict; `city = 'Lahore' OR city = 'Multan'` is generous. When AND and OR mix, use brackets — AND binds tighter than OR, and guessing wrong changes the answer.
- **IN is a tidy OR:** `city IN ('Lahore', 'Multan', 'Karachi')` means "city equals any of these". One line instead of three ORs.
- **BETWEEN is a range, both ends included:** `marks BETWEEN 80 AND 95` includes 80 and 95 themselves.
- **LIKE finds patterns in text:** `%` means "anything (or nothing) here". `name LIKE 'A%'` = names starting with A; `name LIKE '%Khan'` = names ending with Khan; `name LIKE '%ah%'` = names containing "ah" anywhere.

  (Roman Urdu: LIKE mein % wildcard hai — jahan laga do, wahan kuch bhi chalega.)
- **ORDER BY sorts the answer:** `ORDER BY marks DESC` = highest first (descending); `ASC` = lowest first (ascending, the default). Sorting happens on the result — the stored table is never rearranged.
- **LIMIT caps the count:** `LIMIT 3` = first 3 rows of the (sorted) answer. ORDER BY + LIMIT together is the universal "top N" recipe: top 3 students, latest 5 orders, first 10 products.

Code examples

Example 1 — WHERE with AND: one class, high marks

```
SELECT name, class, marks FROM students
WHERE class = '10-A' AND marks >= 88;
```

What it does: Both conditions must hold. Class 10-A has Ali (88) and Sara (92) — both pass the marks bar too. Bilal and Hira are 10-B; Usman is 9-A — all three fail the first condition, whatever their marks.

What you will see:

```
+-----------+--------+-------+
| name      | class  | marks |
+-----------+--------+-------+
| Ali Raza  | 10-A   |    88 |
| Sara Khan | 10-A   |    92 |
+-----------+--------+-------+
2 rows in set
```

Change the `AND` to `OR` in your head: now a row passes if it is 10-A *or* has marks ≥ 88 — Hira (95) joins the answer, giving 3 rows. Same data, different logic, different answer — this is why the AND/OR choice deserves a second of thought.

Example 2 — IN, BETWEEN and LIKE: three sharper sieves

```
SELECT name, city FROM students
WHERE city IN ('Lahore', 'Multan');

SELECT name, marks FROM students
WHERE marks BETWEEN 80 AND 95;

SELECT name FROM students
WHERE name LIKE '%Shah';
```

What it does: Three small queries, three filter tools. IN picks two cities. BETWEEN keeps marks from 80 to 95 — Ali 88, Sara 92, Hira 95 (both ends included, so 95 stays). LIKE '%Shah' finds names *ending* in Shah.

What you will see:

```
+-------------+---------+
| name        | city    |
+-------------+---------+
| Sara Khan   | Lahore  |
| Bilal Ahmed | Multan   |
+-------------+---------+
2 rows in set

+-----------+-------+
| name      | marks |
+-----------+-------+
| Ali Raza  |    88 |
| Sara Khan |    92 |
| Hira Shah |    95 |
+-----------+-------+
3 rows in set

+-----------+
| name      |
+-----------+
| Hira Shah |
+-----------+
1 row in set
```

Three queries, three grids — each tool earning its place. Note Usman (64) fails BETWEEN and Bilal (76) fails it too; the range is honest about its edges.

Example 3 — ORDER BY + LIMIT: the top 3 podium

```
SELECT name, marks FROM students
ORDER BY marks DESC
LIMIT 3;
```

What it does: Sort all five students by marks, highest first, then keep only the first three. This exact pair — ORDER BY a score DESC, LIMIT a small number — powers every "Top 10" list on the internet.

What you will see:

```
+-----------+-------+
| name      | marks |
+-----------+-------+
| Hira Shah |    95 |
| Sara Khan |    92 |
| Ali Raza  |    88 |
+-----------+-------+
3 rows in set
```

Gold, silver, bronze: Hira, Sara, Ali. Remove the LIMIT and you get all five in order (Bilal 76, Usman 64 follow). Change DESC to ASC and the same three slots show the *lowest* three instead — order first, limit second; the pair only makes sense together.

Quick recap

- **WHERE** sieves rows: comparisons, **AND** (both), **OR** (either), **IN** (any of a list), **BETWEEN** (range, ends included), **LIKE** (text patterns with `%`).
- **ORDER BY** sorts the result — `DESC` high to low, `ASC` low to high.
- **LIMIT** caps how many rows return.
- ORDER BY + LIMIT = the universal Top-N recipe.

Common mistakes

- `=` **for text patterns:** `name = '%Shah'` finds literally a name equal to "%Shah" — patterns need LIKE, plain equality needs the exact value.
- **BETWEEN ends confusion:** BETWEEN 80 AND 95 *includes* 80 and 95. If you need strictly inside, write `marks > 80 AND marks < 95` and say so.
  (Roman Urdu: BETWEEN mein dono kinare shamil hote hain — ye aksar bhool jata hai.)
- **LIMIT without ORDER BY:** "first 3 rows" of an unsorted result is whatever order the database felt like — unstable and meaningless. Top-N always sorts first.
- **AND/OR without brackets:** `class = '10-A' OR class = '10-B' AND marks > 90` is read as 10-A (any marks) OR (10-B with marks > 90), because AND binds first. When the meaning matters, bracket it: `(class = '10-A' OR class = '10-B') AND marks > 90`.

Practice

1. Write the query for students whose marks are below 70 or above 90 — predict the names first (check the DB.1 table), then write it with a single WHERE using OR.
2. Find every student whose city name contains the letter 'a' (careful: LIKE is case-insensitive for these names in MySQL's default settings — count your matches by hand first).

3. Write the "bottom 2" query: the two lowest-mark students, lowest first. Then write the query for the single top student of class 10-B only.

Practice quiz

1. Show only students from Lahore.

- SELECT * FROM students WHERE city = 'Lahore';
- SELECT * FROM students WHERE city = Lahore;
- SELECT * FROM students ORDER BY city;
- SELECT Lahore FROM students;

Answer: SELECT * FROM students WHERE city = 'Lahore';

2. Show students with marks above 85.

- SELECT * FROM students WHERE marks > 85;
- SELECT * FROM students WHERE marks = 85;
- SELECT * FROM students WHERE marks < 85;
- SELECT * FROM students HAVING marks > 85;

Answer: SELECT * FROM students WHERE marks > 85;

3. Show students from Lahore OR Multan.

- SELECT * FROM students WHERE city = 'Lahore' OR city = 'Multan';
- SELECT * FROM students WHERE city = 'Lahore' AND city = 'Multan';
- SELECT * FROM students WHERE city = 'Lahore' 'Multan';
- SELECT * FROM students WHERE city IN Lahore;

Answer: SELECT * FROM students WHERE city = 'Lahore' OR city = 'Multan';

4. Show the top 3 students by marks, highest first.

- SELECT * FROM students ORDER BY marks DESC LIMIT 3;
- SELECT * FROM students ORDER BY marks ASC LIMIT 3;
- SELECT TOP students WHERE marks;
- SELECT * FROM students LIMIT 3 ORDER BY marks;

Answer: SELECT * FROM students ORDER BY marks DESC LIMIT 3;

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/sql-where-order-limit · More free tools and notes at https://scopetreasure.com