ScopeTreasure Courses

SQL COURSES · LESSON DB.3 · CODE SCOPE SERIES

CRUD — INSERT / SELECT / UPDATE / DELETE

The four jobs every app does with data: add rows with INSERT, read with SELECT, change with UPDATE, remove with DELETE — and the WHERE that keeps them safe.

Introduction

Every app you have ever used does only four things with its data: **Create** it, **Read** it, **Update** it, **Delete** it — **CRUD**. A school app adds a student (Create), shows the class list (Read), corrects a mark (Update), and removes a student who left (Delete). SQL has exactly one command for each job: `INSERT`, `SELECT`, `UPDATE`, `DELETE`.

This lesson fills our empty `students` table from DB.2 and runs all four operations. One safety rule travels with two of them, and it is the most important sentence in this lesson: **UPDATE and DELETE without WHERE touch every row in the table.**

(Roman Urdu: CRUD = data ke chaar bunyadi kaam. Aur WHERE ke baghair UPDATE/DELETE poori table par chalta hai — ye sabak bhoolna nahi.)

Key points

- **INSERT adds rows:** `INSERT INTO students (columns...) VALUES (...);` — name the columns, then give the values in the same order. Text values wear single quotes; numbers do not.
- **Multi-row INSERT:** one command can add many rows — repeat the `(...)` value groups, separated by commas. Faster and cleaner than five separate commands.
- **SELECT reads:** you already know it from DB.1 — choose the columns, name the table, add `WHERE` to filter. Reading never changes data, so SELECT is the safe command to practise with.
- **UPDATE changes rows that already exist:** `UPDATE students SET marks = 90 WHERE id = 3;` — SET names the new value, **WHERE chooses which rows**. No WHERE = every row gets the new value.
- **DELETE removes rows:** `DELETE FROM students WHERE id = 6;` — again, **WHERE chooses the victim**. No WHERE = the table is emptied (the table itself survives; its rows do not).

  (Roman Urdu: UPDATE aur DELETE mein WHERE bhoolna = poori table badal ya khaali. Hamesha pehle SELECT se check karo ke wohi rows aati hain.)
- **The golden safety habit:** before any UPDATE or DELETE, run a SELECT with the *same* WHERE. If the SELECT shows exactly the rows you meant, run the real command.

Code examples

Example 1 — INSERT: the first three students

```
INSERT INTO students (id, name, class, marks, city)
VALUES (1, 'Ali Raza', '10-A', 88, 'Sahiwal');

INSERT INTO students (id, name, class, marks, city)
VALUES
  (2, 'Sara Khan', '10-A', 92, 'Lahore'),
  (3, 'Bilal Ahmed', '10-B', 76, 'Multan');

SELECT * FROM students;
```

What it does: The first INSERT adds one row; the second adds two rows in a single command — the multi-row form. The final SELECT proves what actually landed.

What you will see:

```
+----+-------------+--------+-------+---------+
| id | name        | class  | marks | city    |
+----+-------------+--------+-------+---------+
|  1 | Ali Raza    | 10-A   |    88 | Sahiwal |
|  2 | Sara Khan   | 10-A   |    92 | Lahore  |
|  3 | Bilal Ahmed | 10-B   |    76 | Multan  |
+----+-------------+--------+-------+---------+
3 rows in set
```

Example 2 — INSERT the rest, SELECT only what you need

```
INSERT INTO students (id, name, class, marks, city)
VALUES
  (4, 'Hira Shah', '10-B', 95, 'Sahiwal'),
  (5, 'Usman Tariq', '9-A', 64, 'Karachi');

SELECT name, city FROM students;
```

What it does: Two more rows complete our five-student table (the same five from DB.1). The SELECT asks for just two columns — in real apps you fetch the columns you need, not `*` every time.

What you will see:

```
+-------------+----------+
| name        | city     |
+-------------+----------+
| Ali Raza    | Sahiwal  |
| Sara Khan   | Lahore   |
| Bilal Ahmed | Multan   |
| Hira Shah   | Sahiwal  |
| Usman Tariq | Karachi  |
+-------------+----------+
5 rows in set
```

Example 3 — UPDATE and DELETE, the safe way

```
INSERT INTO students (id, name, class, marks, city)
VALUES (6, 'Test Student', '9-B', 50, 'Okara');

UPDATE students
SET marks = 55
WHERE id = 6;

DELETE FROM students
WHERE id = 6;

SELECT * FROM students;
```

What it does: A throwaway sixth row is added so the dangerous commands can be practised safely. UPDATE lifts the test student's marks from 50 to 55 — the `WHERE id = 6` guarantees only that row changes. DELETE then removes exactly that row. The real five students are never touched.

What you will see:

```
+----+-------------+--------+-------+----------+
| id | name        | class  | marks | city     |
+----+-------------+--------+-------+----------+
|  1 | Ali Raza    | 10-A   |    88 | Sahiwal  |
|  2 | Sara Khan   | 10-A   |    92 | Lahore   |
|  3 | Bilal Ahmed | 10-B   |    76 | Multan   |
|  4 | Hira Shah   | 10-B   |    95 | Sahiwal  |
|  5 | Usman Tariq | 9-A    |    64 | Karachi  |
+----+-------------+--------+-------+----------+
5 rows in set
```

The test row came and went; the original five are exactly as before. Now imagine the same UPDATE **without** its WHERE line: every student's marks would have become 55. That one line is the difference between a fix

and a disaster.

Quick recap

- **CRUD** = INSERT (create), SELECT (read), UPDATE (change), DELETE (remove) — the four jobs behind every app.
- INSERT names columns, then values in the same order; text wears single quotes.
- UPDATE = `SET` the new value + `WHERE` to choose rows. DELETE = `FROM` + `WHERE`.
- Safety habit: SELECT with the same WHERE first, *then* run the UPDATE or DELETE.

Common mistakes

- **UPDATE or DELETE with no WHERE:** the single most destructive beginner error in SQL. Every row is changed or removed, and there is no undo button. (Transactions in DB.10 are the closest thing to one.)
- **Column/value order mismatch in INSERT:** values land in the columns in the order you listed them. `(name, marks)` with values `(88, 'Ali')` tries to save a number as a name — keep the two lists in the same order.

  (Roman Urdu: Columns aur values ki tartib ek jaisi rakho, warna data ghalat dabbon mein jayega.)
- **Quotes around numbers, or missing quotes around text:** `'88'` is text; `88` is a number. `'Ali Raza'` needs its quotes; `Ali Raza` without them is read as column names and the query fails.
- **Expecting DELETE to remove the table:** DELETE removes *rows*; the empty table stays. Removing the table itself is a different command (`DROP TABLE`) — a much bigger decision.

Practice

1. INSERT two new students of your own invention (ids 6 and 7) in one multi-row command, then SELECT the full table to prove they landed.
2. Bilal's marks were entered wrong: they should be 79, not 76. Write the UPDATE — with its WHERE — then undo your practice change by setting them back to 76, and delete the two students from task 1.
3. Write the SELECT (with WHERE) you would run as a safety check before deleting every student from Karachi. How many rows should it show before you dare run the DELETE?

Practice quiz

1. Show every student, all columns.

- SELECT * FROM students;
- SELECT name FROM students;
- SHOW students;
- GET ALL students;

Answer: SELECT * FROM students;

2. Show only the names of students.

- SELECT name FROM students;
- SELECT * FROM students;
- SELECT names FROM students;
- SHOW name students;

Answer: SELECT name FROM students;

3. Add a new student called Ahmed from Lahore with 90 marks.

- INSERT INTO students (name, city, marks) VALUES ('Ahmed', 'Lahore', 90);
- ADD students ('Ahmed', 'Lahore', 90);
- INSERT students Ahmed Lahore 90;
- UPDATE students SET name = 'Ahmed';

Answer: INSERT INTO students (name, city, marks) VALUES ('Ahmed', 'Lahore', 90);

4. Change the marks of student id 5 to 95.

- UPDATE students SET marks = 95 WHERE id = 5;
- UPDATE students marks = 95;
- SET students marks 95 WHERE id = 5;
- CHANGE students SET marks = 95;

Answer: UPDATE students SET marks = 95 WHERE id = 5;

5. Remove the student with id 9.

- DELETE FROM students WHERE id = 9;
- REMOVE students 9;
- DELETE students id = 9;
- DROP students WHERE id = 9;

Answer: DELETE FROM students WHERE id = 9;

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-crud · More free tools and notes at https://scopetreasure.com