Modifying data

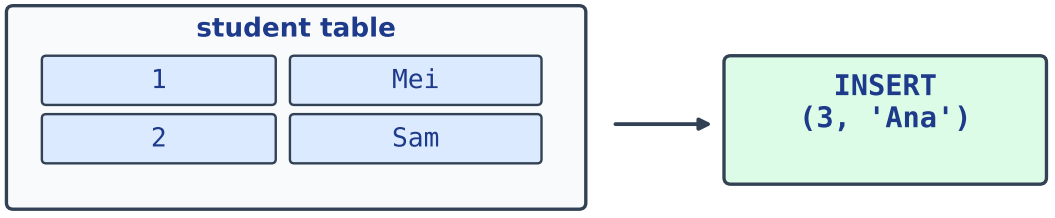
SQL Reference

INSERT

`INSERT INTO ... VALUES ...` adds new rows 行. Name the columns, then give the values in the same order. (Each block below ends with a `SELECT` so you can see the result.)

```
CREATE TABLE student (id INTEGER, name TEXT, score INTEGER);
INSERT INTO student (id, name, score) VALUES (1, 'Mei', 88);
INSERT INTO student VALUES (2, 'Sam', 71);
SELECT * FROM student;
```

- `INSERT INTO t VALUES (...), (...);` adds several rows in one statement.



Exam tip: INSERT INTO table VALUES (...); column order must match

INSERT adds a new row to a table

UPDATE

`UPDATE ... SET ... WHERE ...` changes existing rows. **Always** add `WHERE`, or every row changes.

```
CREATE TABLE student (id INTEGER, name TEXT, score INTEGER);
INSERT INTO student VALUES (1, 'Mei', 88), (2, 'Sam', 71);
UPDATE student SET score = 75 WHERE name = 'Sam';
SELECT * FROM student;
```

DELETE

`DELETE FROM ... WHERE ...` removes rows. Without `WHERE` it empties the whole table 表.

```
CREATE TABLE student (id INTEGER, name TEXT, score INTEGER);
INSERT INTO student VALUES (1, 'Mei', 88), (2, 'Sam', 71);
DELETE FROM student WHERE score < 80;
SELECT * FROM student;
```

Common mistakes

- `UPDATE` and `DELETE` **without** a `WHERE` change every row — always add the `WHERE`.
- In `INSERT`, the values must line up with the column list in order and type.
- Test a risky `DELETE` first as a `SELECT` with the same `WHERE`.