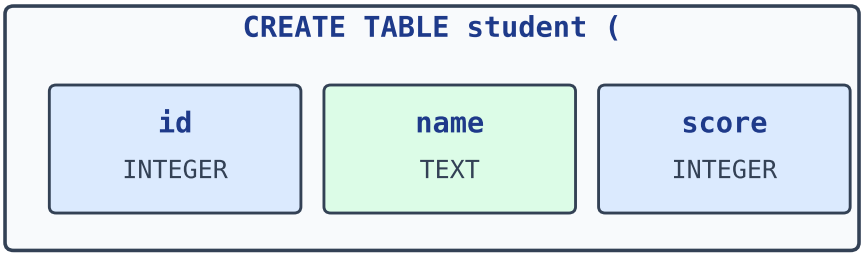
# Defining tables

SQL Reference

## CREATE TABLE & data types

`CREATE TABLE` defines a table's schema : the column names and their data types . The main SQLite types are `INTEGER`, `TEXT`, and `REAL` (a floating-point number: fast, but not always exact, so `0.1 + 0.2 = 0.3` is false).

```
CREATE TABLE student (id INTEGER, name TEXT, score REAL);
INSERT INTO student VALUES (1, 'Mei', 88.5);
SELECT * FROM student;
```




Exam tip: name columns and types first; PRIMARY KEY marks the unique id

*CREATE TABLE names columns and their types*

## Keys & constraints

A constraint is a rule on a column: `PRIMARY KEY` (a unique id), `NOT NULL` (must have a value), `UNIQUE`, and `DEFAULT` (a fallback value).

```
CREATE TABLE student (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    score INTEGER DEFAULT 0
);
INSERT INTO student (id, name) VALUES (1, 'Mei');
SELECT * FROM student;
```

Declare a foreign key with `REFERENCES`. In SQLite, enable enforcement for each connection with `PRAGMA foreign_keys = ON;` before starting a transaction. A declaration alone does not guarantee enforcement.

```
PRAGMA foreign_keys = ON;
CREATE TABLE class (id INTEGER PRIMARY KEY, name TEXT);
CREATE TABLE student (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    class_id INTEGER REFERENCES class(id)
);
INSERT INTO class VALUES (1, 'Maths');
INSERT INTO student VALUES (1, 'Mei', 1);
SELECT s.name, c.name AS class
FROM student s INNER JOIN class c ON s.class_id = c.id;
```

- The long form is `FOREIGN KEY (class_id) REFERENCES class(id)` on its own line.

## ALTER TABLE

`ALTER TABLE ... ADD COLUMN ...` changes the schema of a table that already exists. A `DEFAULT` fills the new column in the old rows.

```
CREATE TABLE student (id INTEGER, name TEXT);
INSERT INTO student VALUES (1, 'Mei');
ALTER TABLE student ADD COLUMN score INTEGER DEFAULT 0;
SELECT * FROM student;
```

- `ALTER TABLE student RENAME TO pupil;` renames the whole table.
- `DROP TABLE student;` deletes the table completely – structure **and** data.

### Common mistakes

- Declare a suitable type for each column. SQLite permits omitted type names and normally uses flexible typing; a type declaration alone does not validate every stored value.
- Design a primary key to be unique and non-NULL. In ordinary SQLite tables, some primary-key declarations still permit NULL; use `NOT NULL` explicitly for a text key. `INTEGER PRIMARY KEY` has special row-id behavior: inserting NULL allocates an integer id.
- With foreign-key enforcement enabled, a non-NULL foreign-key value must match a referenced key. NULL is allowed unless the column also has `NOT NULL`.