ScopeTreasure Courses

SQL COURSES · LESSON DB.2 · CODE SCOPE SERIES

Pehli Table Banana — CREATE + Data Types

Build your first table: CREATE TABLE, the main data types (INT, VARCHAR, DECIMAL, DATE, BOOLEAN), and reading the DESCRIBE result.

Introduction

In DB.1 we *read* a table. Now we **build** one. Before a single student can be saved, the database must know the table's shape: what the columns are called, and what kind of value each column accepts. That shape is written once, in a `CREATE TABLE` command — and every column gets a **data type**, the rule that says "this box holds numbers only" or "this box holds short text".

Get the types right here and a whole family of bugs never happens: marks stored as text cannot be averaged, and a name stored in a too-small box gets cut. This lesson builds the same `students` table the series uses, then reads its blueprint back with `DESCRIBE`.

(Roman Urdu: Table banana = pehle dabbon ki shape tay karna. Type ghalat likha to data andar jate hi bigadne lagega.)

Key points

- **CREATE TABLE writes the blueprint:** `CREATE TABLE students (...columns...);` — column name first, then its type, one column per line, separated by commas (no comma after the last one).
- **INT — whole numbers:** marks, age, counts, ids. No decimals. `88` is an INT; `88.5` is not.
- **VARCHAR(n) — short text with a limit:** names, cities, class labels. `VARCHAR(50)` means "text up to 50 characters". The limit is a promise you choose — pick a size that fits real data with room to spare.
- **TEXT — long text:** addresses, descriptions, notes. Use it when the text has no small natural limit; use VARCHAR when there is one.
- **DECIMAL(p, s) — exact money numbers:** `DECIMAL(10,2)` holds up to 10 digits, 2 of them after the point: `1450.50`. Money must be DECIMAL, never a floating type — floats round silently, and paisa-level rounding in accounts is a real bug.

  (Roman Urdu: Paison ke liye hamesha DECIMAL — float chup chaap gol kar deta hai.)
- **DATE — a calendar day:** `2026-10-05`, always year-month-day. Dates stored as DATE can be sorted and compared correctly; dates stored as text cannot.
- **BOOLEAN — true/false:** stored as 1/0 in MySQL. Perfect for flags like `is_paid`.
- **DESCRIBE reads the blueprint back:** `DESCRIBE students;` (or `DESC students;`) shows every column, its type, and more. It is your first debugging tool: when a query fails, check the real shape before guessing.

Code examples

Example 1 — Create the students table

```
CREATE TABLE students (
  id INT,
  name VARCHAR(50),
  class VARCHAR(10),
  marks INT,
  city VARCHAR(50)
);
```

What it does: Five columns, each with a type matched to its job: `id` and `marks` are whole numbers; `name` and `city` are text up to 50 characters; `class` is a short label like `10-A`, so 10 characters is plenty. The table is empty — a blueprint, no data yet (data arrives in DB.3).

What you will see:

```
Query OK, 0 rows affected
```

That plain line is the success message: the table now exists. No grid appears, because CREATE is about structure, not data.

Example 2 — Read the blueprint back with DESCRIBE

```
DESCRIBE students;
```

What it does: Asks the database to show the shape it just stored — our first quality check that the blueprint matches what we intended.

What you will see:

```
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int         | YES  |     | NULL    |       |
| name  | varchar(50) | YES  |     | NULL    |       |
| class | varchar(10) | YES  |     | NULL    |       |
| marks | int         | YES  |     | NULL    |       |
| city  | varchar(50) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
5 rows in set
```

Read it row by row: `Field` is the column name, `Type` is the rule we chose. The `Null` column says YES everywhere — for now any column may be left empty; DB.6 and DB.7 will tighten these rules with keys and

constraints.

Example 3 — A second table, and the type-size trap

```
CREATE TABLE fees (
  id INT,
  student_id INT,
  amount DECIMAL(10,2),
  paid_on DATE,
  is_paid BOOLEAN
);

INSERT INTO fees (id, student_id, amount, paid_on, is_paid)
VALUES (1, 1, 5000.00, '2026-09-05', TRUE);
```

What it does: The `fees` table shows the remaining types in action: money as `DECIMAL(10,2)`, a real calendar day as `DATE`, a yes/no flag as `BOOLEAN`. The INSERT (full treatment in DB.3) drops one row in to prove the shape works.

What you will see:

```
Query OK, 1 row affected
```

Now the trap: try saving a student into a test column made as `name VARCHAR(5)` with the value `'Bilal Ahmed'` — MySQL refuses: `ERROR 1406: Data too long for column 'name'`. The type's size is a real wall. Choose sizes for real data, not for the shortest name you can think of.

Quick recap

- `CREATE TABLE` writes the blueprint once; data comes later.
- Match the type to the job: **INT** whole numbers, **VARCHAR(n)** short text, **TEXT** long text, **DECIMAL** money, **DATE** calendar days, **BOOLEAN** true/false.
- `DESCRIBE table;` reads the blueprint back — your first debugging tool.
- A VARCHAR size is a wall: too small, and real data gets refused.

Common mistakes

- **Marks or money as text:** if `marks` is VARCHAR, the database cannot average or sort it as a number — `'100'` sorts before `'88'` as text. Numbers that will ever be calculated belong in numeric types.
- **Float for money:** `FLOAT` / `DOUBLE` round silently. Fees, prices and balances are `DECIMAL` — always.
  (Roman Urdu: Float paisa gol kar deta hai; hisaab mein ye chhoti ghalti badi banti hai.)
- **Comma after the last column:** `CREATE TABLE` lists end without a trailing comma. A stray comma is the most common CREATE error for beginners.

- **Dates as text:** `'05-10-2026'` typed into a VARCHAR cannot be compared or sorted as a date. If it is a day, its type is DATE, and its format is year-month-day.

Practice

1. Write the CREATE TABLE for a `books` table: id (whole number), title (text up to 100), author (text up to 50), price (money), published_on (a calendar day). Then DESCRIBE it and check every type.
2. A shop wants a `products` table with name, price and stock count. Choose the type for each column and write one line of reasoning per choice.
3. Spot the three type mistakes: `phone VARCHAR(3)`, `price FLOAT`, `birthday VARCHAR(20)`. Rewrite the table with correct types.

Practice quiz

1. Which statement writes a table's blueprint?

- INSERT INTO students (...);
- CREATE TABLE students (...columns...);, column name first, then its type, one per line, no comma after the last one
- DESCRIBE students;
- SELECT * FROM students;

Answer: CREATE TABLE students (...columns...);, column name first, then its type, one per line, no comma after the last one

2. Which type must money use, and why?

- FLOAT, because it is fastest
- DECIMAL, because floats round silently and paisa-level rounding in accounts is a real bug
- VARCHAR, so the currency symbol fits
- INT, because money is always whole

Answer: DECIMAL, because floats round silently and paisa-level rounding in accounts is a real bug

3. How is a DATE value written, and why not as text?

- 05-10-2026, day first, as text is fine
- 2026-10-05, always year-month-day; dates stored as DATE can be sorted and compared correctly, dates stored as text cannot
- 10/05/2026, either order works
- As a number of days only

Answer: 2026-10-05, always year-month-day; dates stored as DATE can be sorted and compared correctly, dates stored as text cannot

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