Inserting & Reading Data

AiCE Software Engineering Foundations — MySQL Programming

The books table exists, but it's empty — nothing but headers. This lesson covers the first half of CRUD: Create, using INSERT INTO to add real books, and Read, using SELECT to bring that data back out, filtered, sorted, and limited exactly the way you need it.

1. The Shape of INSERT INTO

To add data to a table, you use the INSERT INTO command, naming the table, the columns you're filling in, and the values to put in them. Let's add our first book:

```
INSERT INTO books (title, author, genre, price, quantity,
date_added)
VALUES ('Things Fall Apart', 'Chinua Achebe', 'Fiction',
15.99, 25, '2025-01-10');
```

Press Enter and you will see Query OK, 1 row affected. Notice id was never provided — MySQL generated it automatically, because of AUTO_INCREMENT.

2. Inserting Multiple Rows at Once

Adding one book at a time works, but MySQL lets you insert several rows in a single statement — useful when you have a batch of data ready to go. Let's add a few more books, so there's enough data to work with:

```
INSERT INTO books (title, author, genre, price, quantity,
date_added)
VALUES
    ('Half of a Yellow Sun', 'Chimamanda Ngozi Adichie',
'Fiction', 18.50, 30, '2025-01-12'),
    ('The Alchemist', 'Paulo Coelho', 'Fiction', 12.00, 40,
'2025-01-15'),
    ('Sapiens', 'Yuval Noah Harari', 'Non-Fiction', 22.75, 15,
'2025-01-20'),
```

```
    ('Atomic Habits', 'James Clear', 'Self-Help', 20.00, 50,
'2025-02-01'),
    ('Purple Hibiscus', 'Chimamanda Ngozi Adichie', 'Fiction',
14.25, 20, '2025-02-05');
```

This produces Query OK, 5 rows affected. Combined with the first insert, aice_books now holds six books in total.

3. Reading Data With SELECT

To view data, you use the SELECT command. To bring back every column of every row:

```
SELECT * FROM books;
```

This returns all six books, each with its auto-generated id. The asterisk (*) means "every column" — useful for a quick look, but real applications usually ask for only what they need.

You can select just the columns you care about instead:

```
SELECT title, author, price FROM books;
```

This returns the same six rows, but only three columns wide — no id, genre, quantity, or date_added.

4. Filtering Rows With WHERE

SELECT * hands back everything. In a real application, users search — they want the Fiction books, or the books under a certain price. The WHERE clause filters rows so only the ones matching a condition come back. Let's find only the Fiction books:

```
SELECT * FROM books WHERE genre = 'Fiction';
```

This returns four books — every row where genre is exactly 'Fiction'. Now find books that cost less than 15:

```
SELECT title, price FROM books WHERE price < 15;
```

This returns two books. WHERE works with text (using =) and with numbers (using =, <, >, and other comparison operators), and it's the same mechanism behind every search box you've ever used in an app.

> **WHERE Is How Applications Search**
> *Every time you filter a shopping site by category, or search for a username, an application is running a SELECT with a WHERE clause behind the scenes — the same clause you just used.*

5. Sorting Results With ORDER BY

Without instruction, MySQL doesn't guarantee any particular row order. ORDER BY sorts your results by a column you choose. Let's sort books from cheapest to most expensive:

```
SELECT title, price FROM books ORDER BY price ASC;
```

Or most expensive first, by switching ASC to DESC:

```
SELECT title, price FROM books ORDER BY price DESC;
```

6. Limiting Results With LIMIT

Sometimes you only want a handful of rows back — the top result, the first page of a list. LIMIT caps how many rows come back. Let's see just the 3 cheapest books:

```
SELECT title, price FROM books ORDER BY price ASC LIMIT 3;
```

This returns exactly three rows, starting from the cheapest — ORDER BY decides the order, and LIMIT decides how many of that ordered list actually come back.

Key Takeaways

INSERT INTO adds one row, or several rows at once, to a table — and MySQL fills in any AUTO_INCREMENT or DEFAULT column for you. SELECT reads data back, either every column with *, or only the ones you name. WHERE filters which rows come

back, ORDER BY controls what order they arrive in, and LIMIT caps how many of them you actually see. Together, these are the C and R in CRUD. The next lesson covers the other half — Update and Delete — including the one habit that matters more than any syntax: always use WHERE.