To implement a **many-to-many relationship** in **MySQL** (without using Django), you'll need **three tables**:

1. A table for `Author`
2. A table for `Book`
3. A **junction table** (also called a **join table** or **association table**) to connect `authors` and `books`.

□ Your Django Models:

```
class Author(models.Model):
    name = models.CharField(max_length=100)

class Book(models.Model):
    title = models.CharField(max_length=100)
    authors = models.ManyToManyField(Author)
```

□ Equivalent MySQL Schema

Here's how you would implement the same structure in **pure MySQL**:

1. `Author` Table

```
CREATE TABLE Author (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);
```

2. `Book` Table

```
CREATE TABLE Book (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL
);
```

3. `BookAuthor` Table (the join table)

This table creates the **many-to-many relationship** between `Book` and `Author`.

```
CREATE TABLE BookAuthor (
    book_id INT NOT NULL,
    author_id INT NOT NULL,
    PRIMARY KEY (book_id, author_id),
    FOREIGN KEY (book_id) REFERENCES Book(id) ON DELETE CASCADE,
    FOREIGN KEY (author_id) REFERENCES Author(id) ON DELETE CASCADE
);
```

□ Explanation:

Table	Purpose
`Author`	Stores author records
`Book`	Stores book records
`BookAuthor`	Links authors to books (many-to-many)

- `BookAuthor` has a **composite primary key** (`book_id`, `author_id`) to prevent duplicates.
- `FOREIGN KEY` constraints ensure that you cannot link to a non-existent book or author.
- `ON DELETE CASCADE` ensures that when a book or author is deleted, the corresponding links are also removed.

□ Example Usage

□ Insert Authors

```
INSERT INTO Author (name) VALUES ('Author A'), ('Author B');
```

□ Insert Books

```
INSERT INTO Book (title) VALUES ('Book 1'), ('Book 2');
```

□ Link Authors to Books

```
-- Link 'Book 1' to both authors
INSERT INTO BookAuthor (book_id, author_id) VALUES (1, 1), (1, 2);

-- Link 'Book 2' to 'Author A' only
INSERT INTO BookAuthor (book_id, author_id) VALUES (2, 1);
```

□ Query Examples

Get all authors of a specific book

```
SELECT a.name
FROM Author a
JOIN BookAuthor ba ON a.id = ba.author_id
WHERE ba.book_id = 1;
```

Get all books written by a specific author

```
SELECT b.title
FROM Book b
JOIN BookAuthor ba ON b.id = ba.book_id
WHERE ba.author_id = 1;
```