**SVKM’s NMIMS**

**School of Technology Management & Engineering (Indore Campus)**

**Computer Engineering Department (B Tech/MBATech CE and B Tech AIDS Sem IV)**

**Database Management System**

**Project Report**

|  |  |  |
| --- | --- | --- |
| Program B-TECH CE A |  | |
| Semester- 4th |  | |
| Name of the Project: BOOKSTORE |  | |
|  | | |
| Details of Project Members |  |  |
| Batch - 02 | Roll No. - D087 | Name- STUTI GUPTA |
|  | Roll No. - D056 | Name- NIYATI BANSAL |
|  | Roll No. -D052 | Name- NALIN GUPTA |
|  |  |  |
| Date of Submission: 11-04-25 | | |

**Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| Roll No. | Name: | Contribution |
| D087 , D052 | STUTI GUPTA , NALIN GUPTA | SQL QUERIES, NORMALIZATION |
| D056 | NIYATI BANSAL | ER DIAGRAM , GUI |

**Github link of your project:**

**Note:**

1. Create a readme file if you have multiple files
2. All files must be properly named (Example:R004\_DBMSProject)
3. Submit all relevant files of your work ( Report, all SQL files, Any other files)
4. **Plagiarism is highly discouraged (Your report will be checked for plagiarism)**

**Rubrics for the Project evaluation:**

|  |  |
| --- | --- |
| First phase of evaluation:  Innovative Ideas (5 Marks)  Design and Partial implementation (5 Marks) | 10 marks |
| Final phase of evaluation  Implementation, presentation and viva, Self-Learning and Learning Beyond classroom | 10 marks |

**Project Report**

**Selected Topic**

**by**

**Stuti Gupta, Roll number: D087**

**Niyati Bansal, Roll number: D056**

**Nalin Gupta, Roll number: D052**

**Course: DBMS**

**AY: 2024-25**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline |  |
| **2** | Components of Database Design |  |
| **3** | Entity Relationship Diagram |  |
| **4** | Relational Model |  |
| **5** | Normalization |  |
| **6** | SQL Queries |  |
| **7** | Learning from the Project |  |
| **8** | Project Demonstration |  |
| **9** | Self-learning beyond classroom |  |
| **10** | Learning from the project |  |
| **8** | Challenges faced |  |
| **9** | Conclusion |  |

**Storyline**

The project titled "Bookstore" aims to create a comprehensive database system for managing the operations of a bookstore. This includes managing books, authors, customers, orders, and inventory. The goal is to enable easy tracking of book availability, order processing, and customer management, while ensuring the database is well-designed, normalized, and efficiently queried.

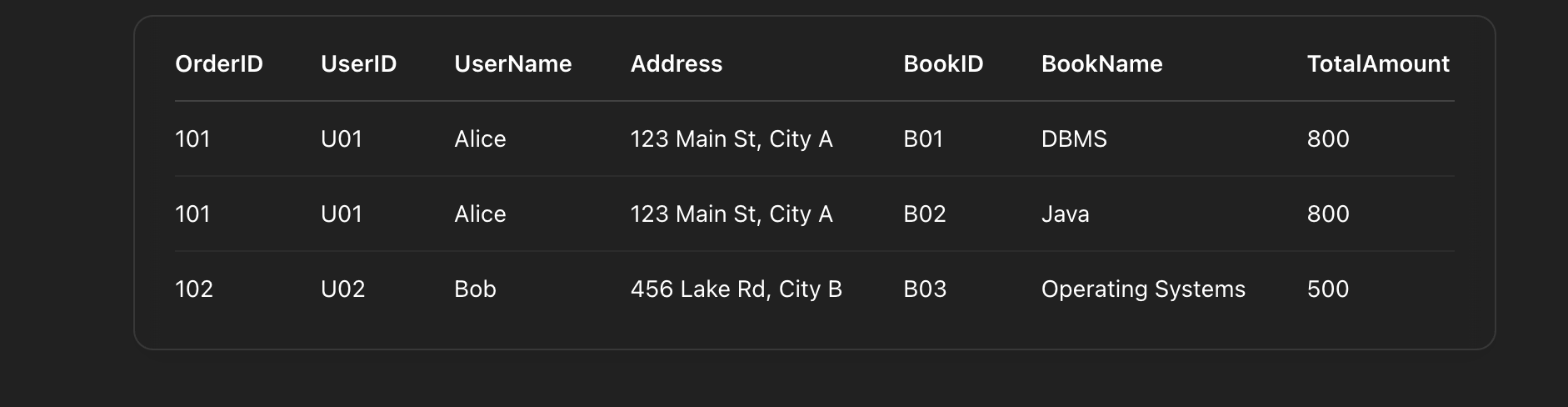
The storyline follows the typical functioning of a bookstore, where customers browse books by genre, author, or title, place orders, and receive invoices. The administrator or store staff manages book entries, monitors stock levels, and handles customer interactions.

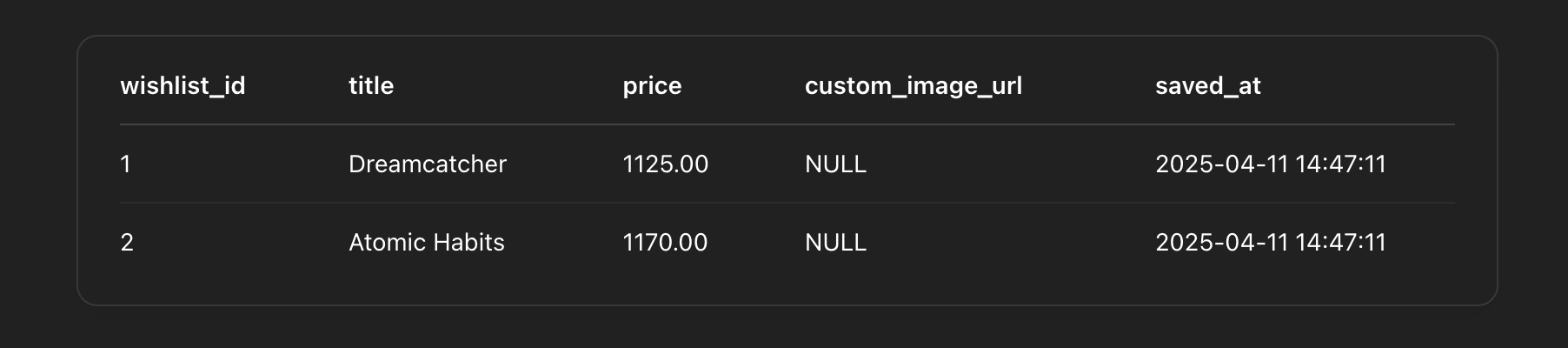
**Entity Relationship Diagram**

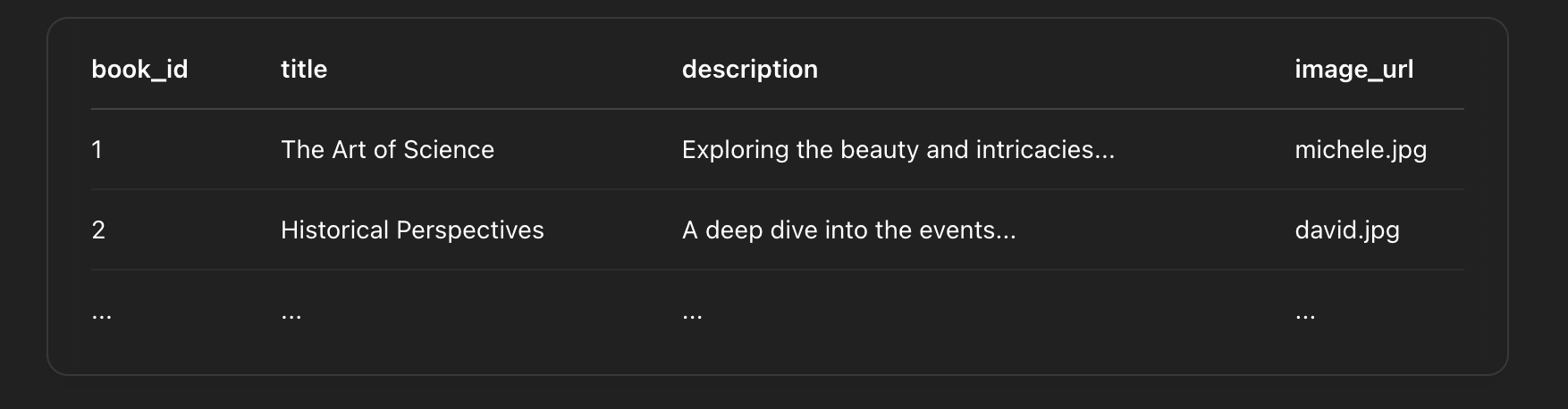


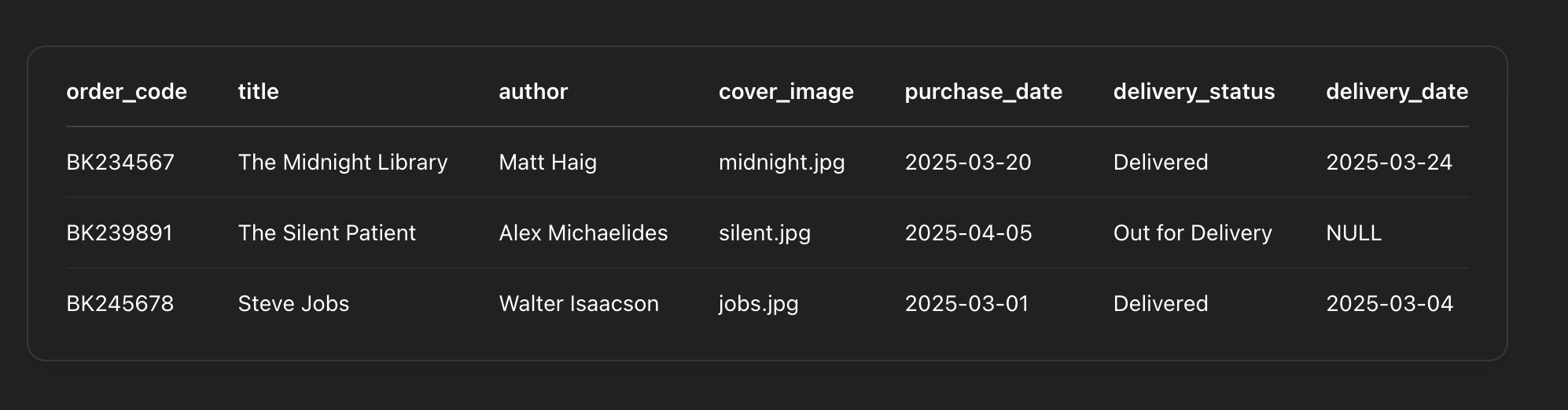
**Normalization**

**First Normal Form (1NF):** All tables have atomic values and unique columns.

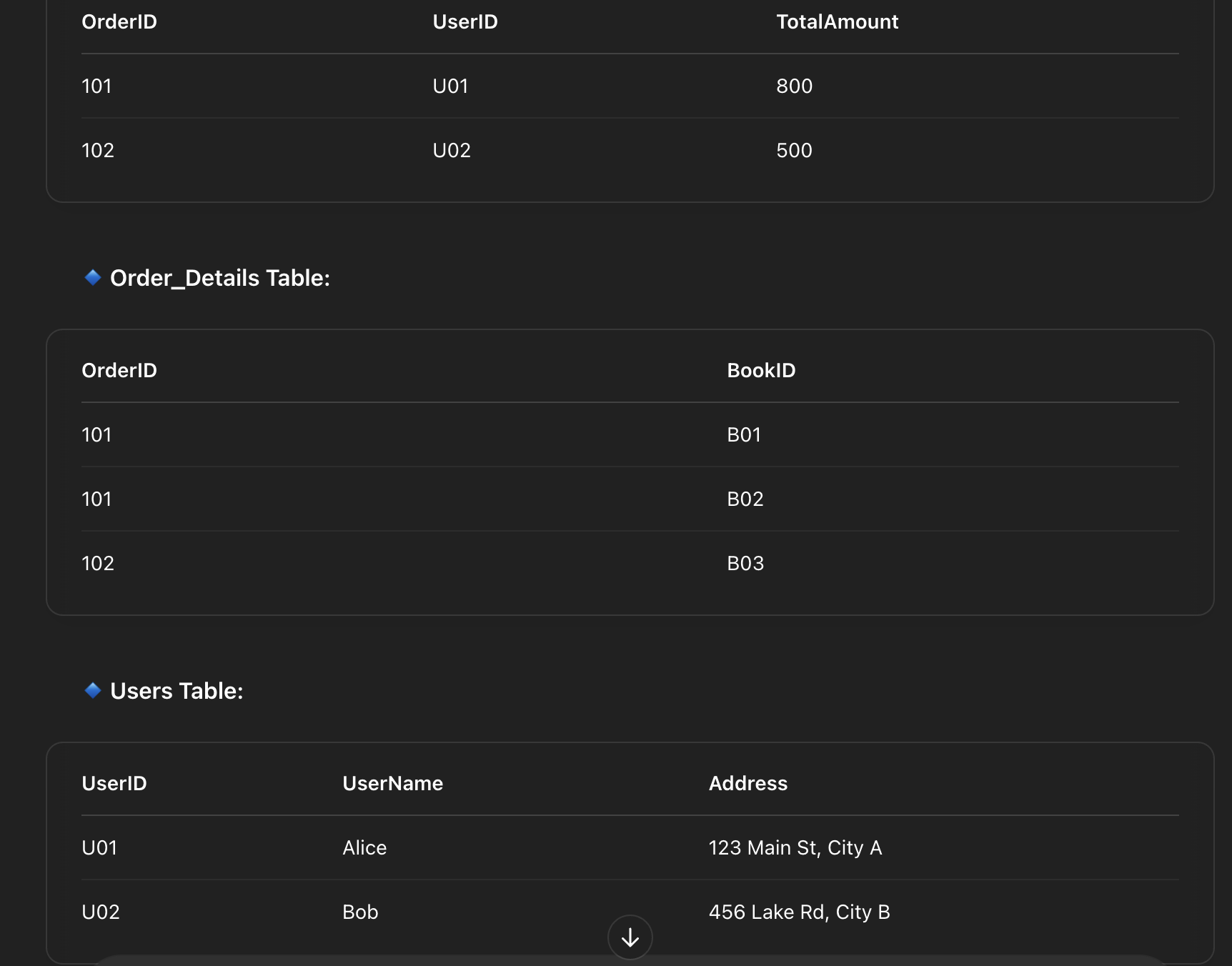


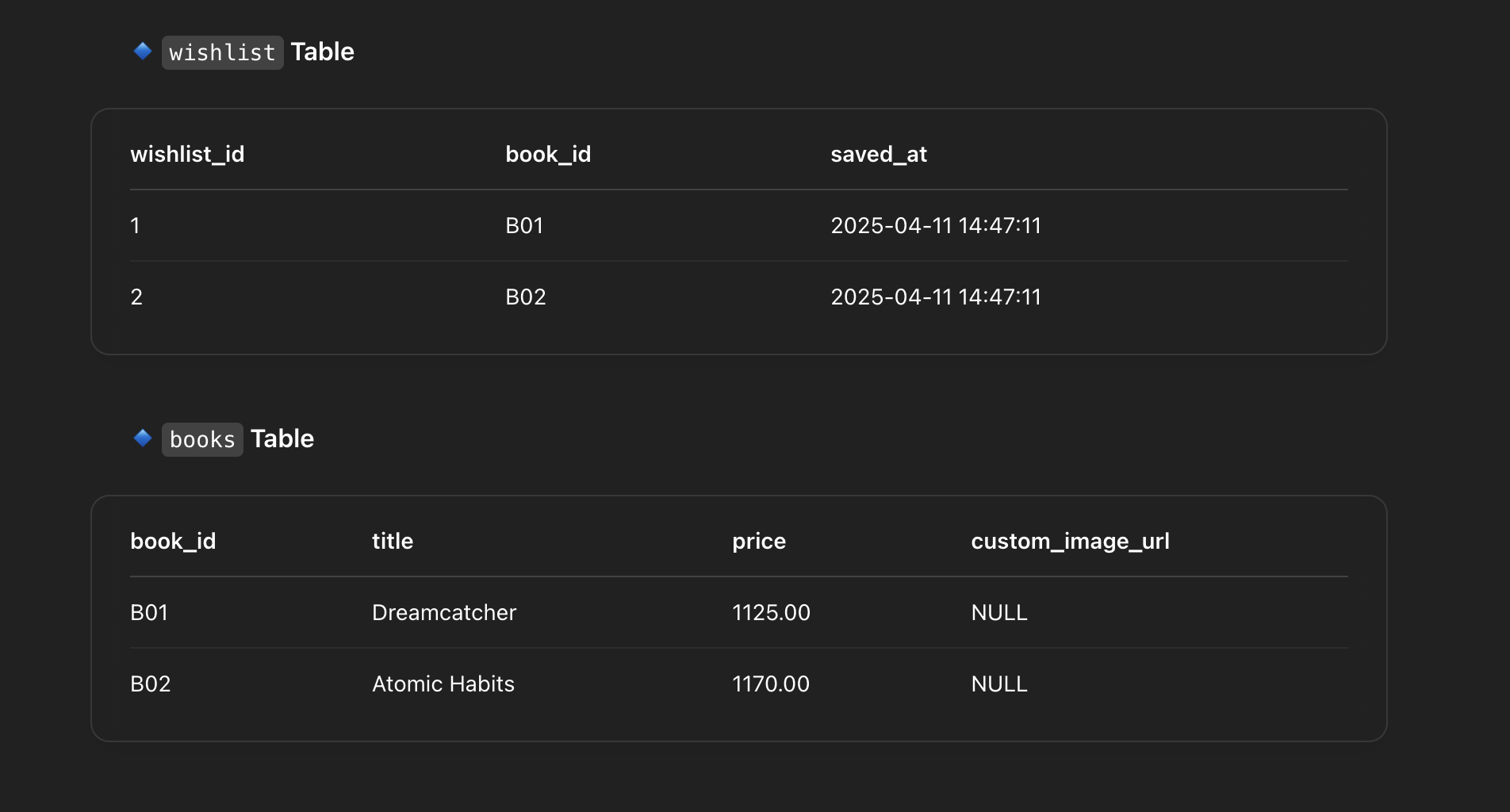


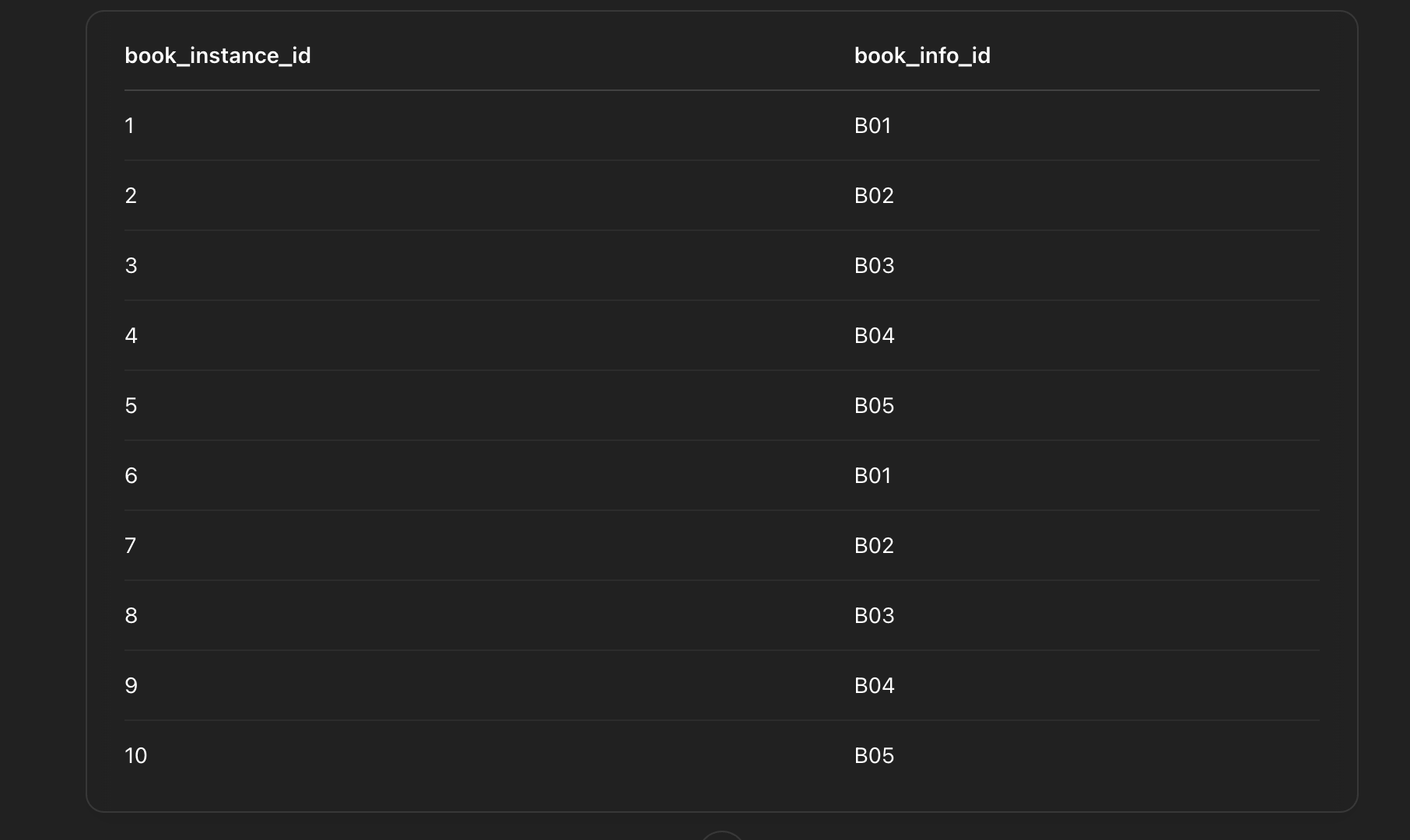


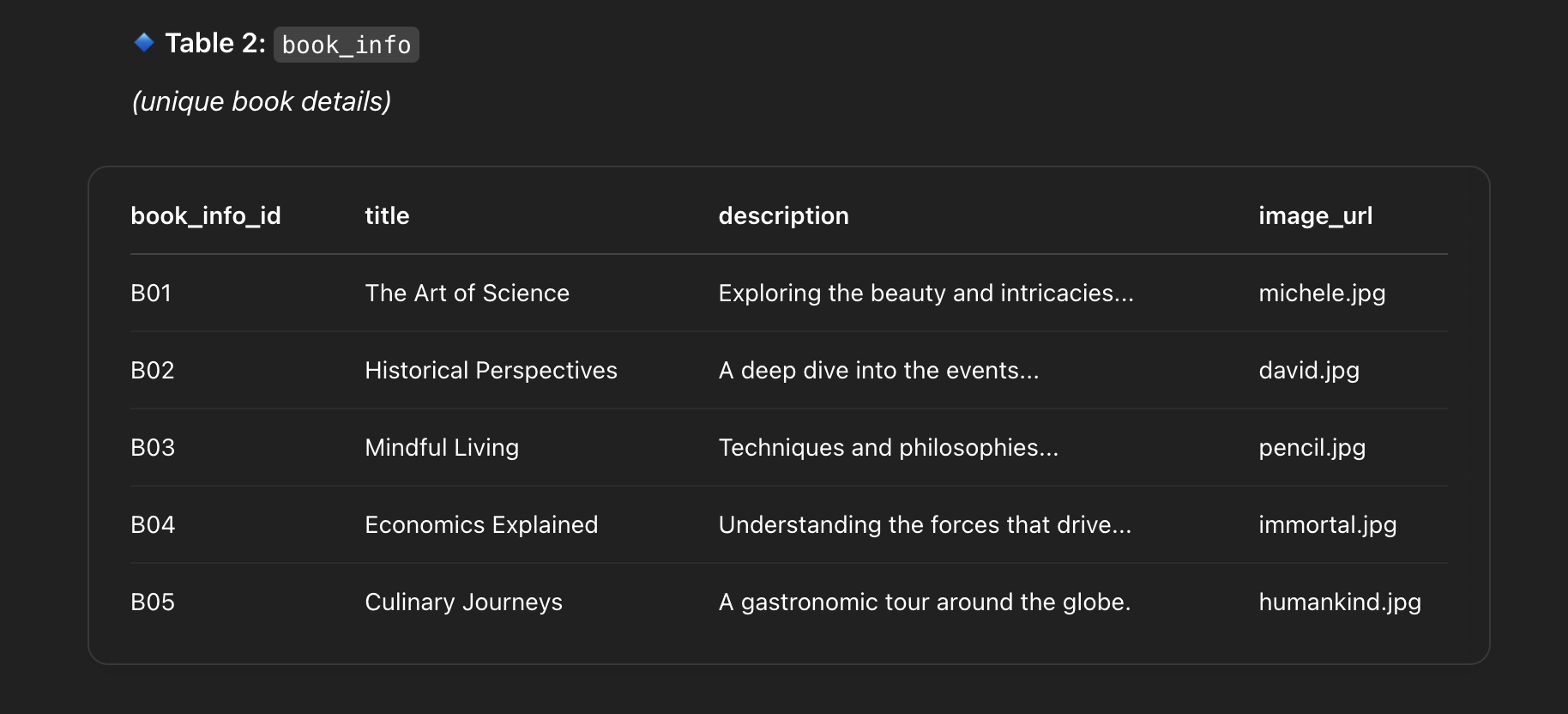
****

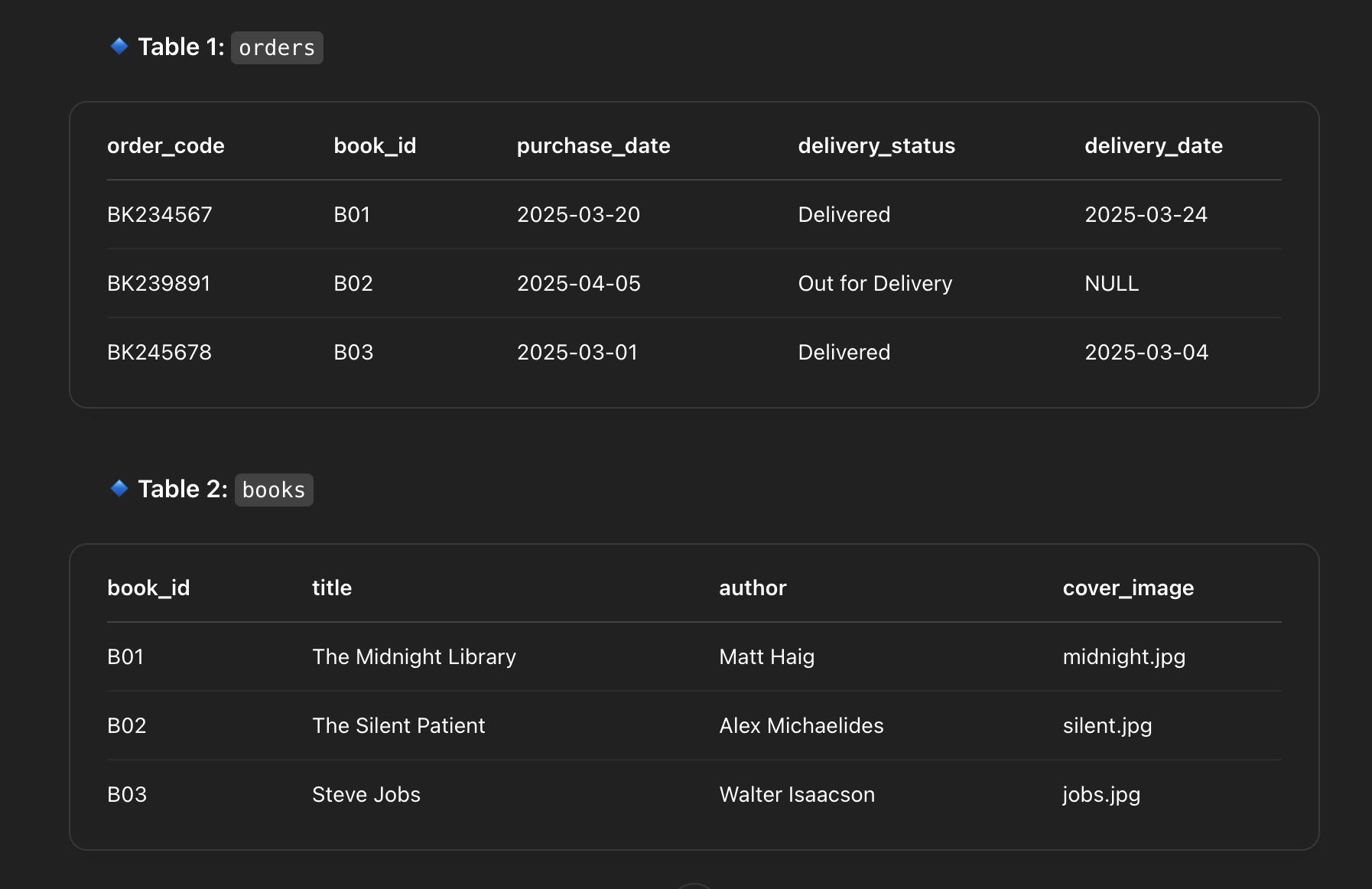
**Second Normal Form (2NF):** Partial dependencies removed. Composite keys are used appropriately in junction.



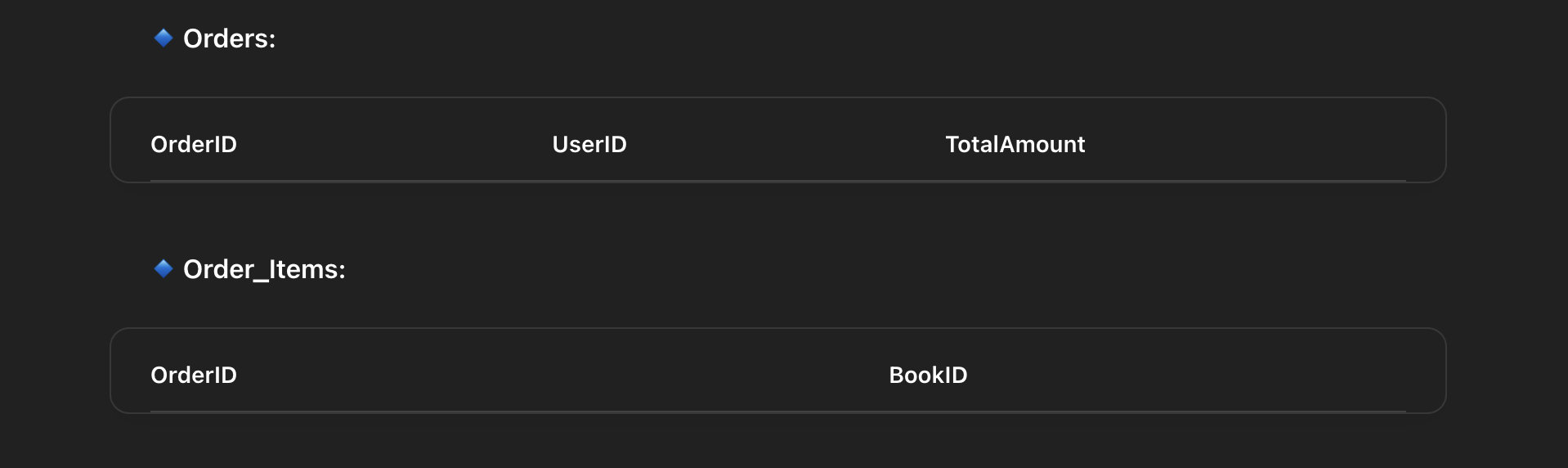


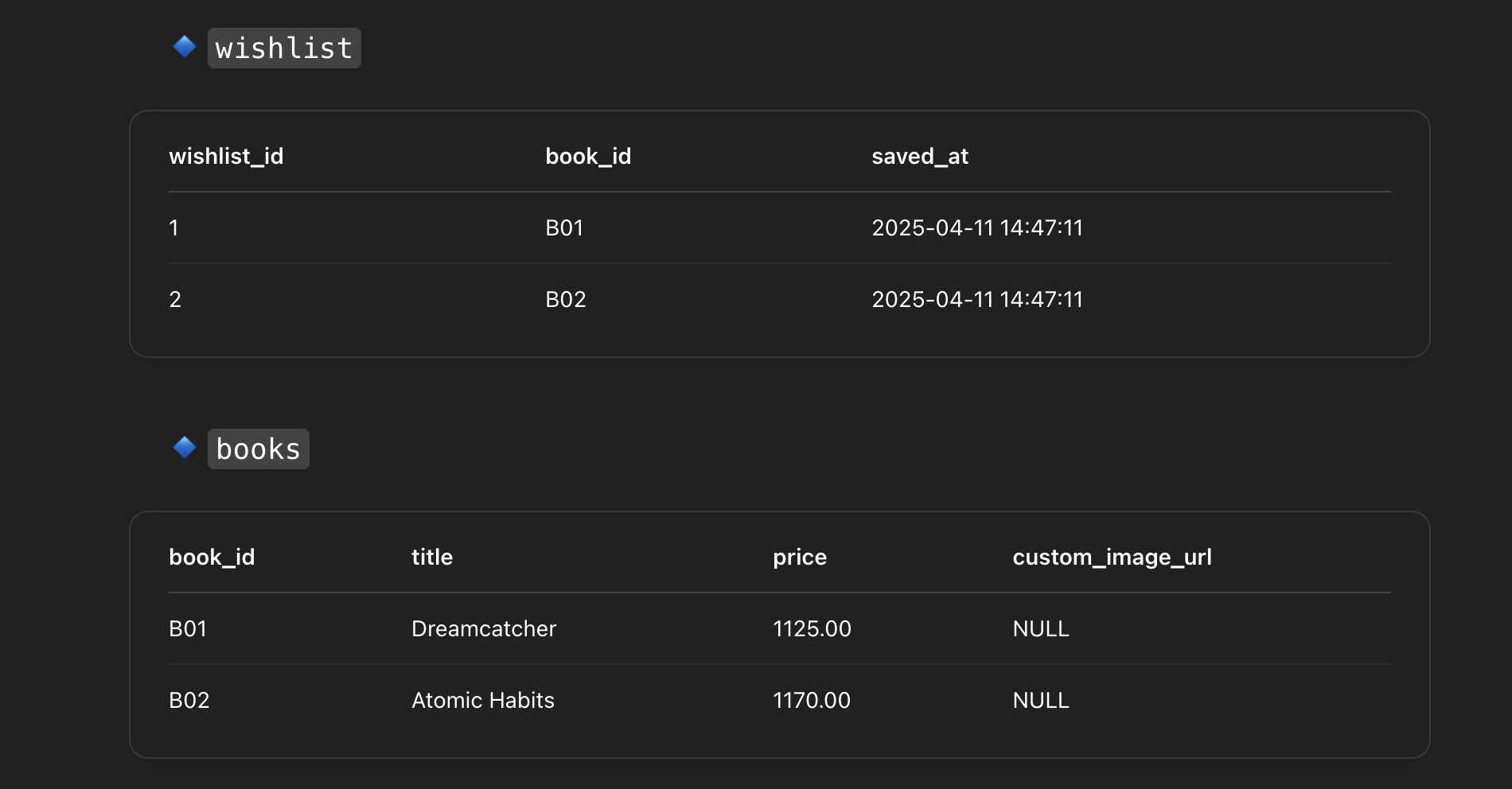


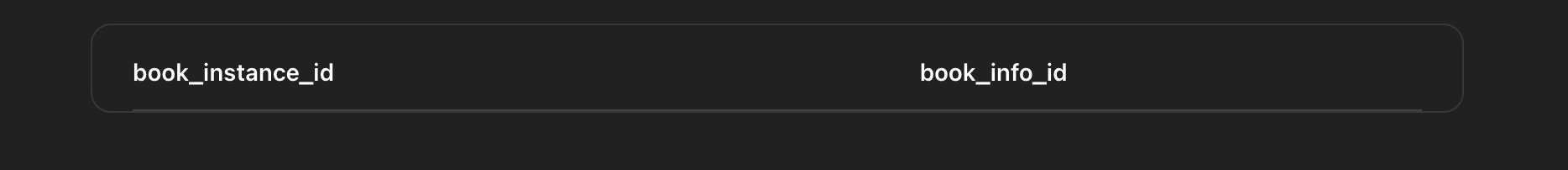


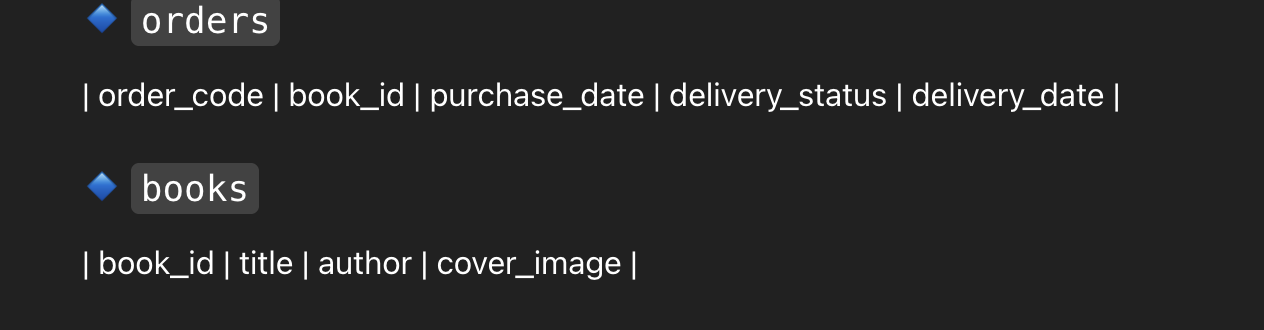


**Third Normal Form (3NF):** No transitive dependencies. Each non-key attribute depends only on the key.









**SQL Queries**

**Code:**

CREATE DATABASE bookstore;

USE bookstore;

-- Create books table

CREATE TABLE books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

image\_url VARCHAR(500)

);

-- Create users table

CREATE TABLE users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(100),

email VARCHAR(150)

);

-- Create wishlist table

CREATE TABLE wishlist (

wishlist\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

book\_id INT,

saved\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (book\_id) REFERENCES books(book\_id)

);

-- Create orders table

CREATE TABLE orders (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

total\_amount DECIMAL(10, 2),

payment\_method VARCHAR(100),

coupon\_code VARCHAR(50),

giftcard\_code VARCHAR(50),

ordered\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

-- Create order\_items table

CREATE TABLE order\_items (

item\_id INT PRIMARY KEY AUTO\_INCREMENT,

order\_id INT,

book\_id INT,

price DECIMAL(10, 2),

FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (book\_id) REFERENCES books(book\_id)

);

INSERT INTO books (title, price, image\_url) VALUES

('The Rising Sun', 1299, 'sun.jpg'),

('Into the Wild', 1499, 'into.jpg'),

('Echoes of Silence', 950, 'echoes.jpg'),

('Dreamcatcher', 1125, 'dream.jpg'),

('Mystic River', 1345, 'river.jpg'),

('The Gene Whisper', 1120, 'gene.jpg'),

('Inside the Mind', 1100, 'brian.jpg'),

('Daring Greatly', 1145, 'daring.jpg'),

('Anne’s Diary', 1085, 'anne.jpg'),

('Tell the Wolves I''m Home', 1150, 'brunt.jpg');

-- add more if needed

SHOW TABLES;

USE booknook;

-- Create books table

CREATE TABLE books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

image\_url VARCHAR(500)

);

-- Create users table

CREATE TABLE users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(100),

email VARCHAR(150)

);

-- Create wishlist table

CREATE TABLE wishlist (

wishlist\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

book\_id INT,

custom\_image\_url VARCHAR(500),

saved\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (book\_id) REFERENCES books(book\_id)

);

-- Insert sample user

INSERT INTO users (username, email) VALUES ('nalin', 'nalin@example.com');

-- Insert sample books

INSERT INTO books (title, price, image\_url) VALUES

('Dreamcatcher', 1125, 'dream.jpg'),

('Into the Wild', 1499, 'into.jpg'),

('Atomic Habits', 1170, 'atomic.jpg');

INSERT INTO wishlist (user\_id, book\_id) VALUES (1, 1), (1, 3);

SELECT w.wishlist\_id, b.title, b.price, w.custom\_image\_url, w.saved\_at

FROM wishlist w

JOIN books b ON w.book\_id = b.book\_id

WHERE w.user\_id = 1;

DELETE FROM wishlist

WHERE user\_id = 1 AND book\_id = 1;

USE booknooc;

-- Table for users

CREATE TABLE users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(100),

email VARCHAR(150)

);

-- Table for books (in your library)

CREATE TABLE books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255),

author VARCHAR(255),

pdf\_path VARCHAR(255),

cover\_image VARCHAR(255)

);

-- Mapping table: books owned by a user

CREATE TABLE user\_library (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

book\_id INT,

added\_on TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (book\_id) REFERENCES books(book\_id)

);

-- Add a user

INSERT INTO users (username, email)

VALUES ('nalin', 'nalin@example.com');

-- Add some books

INSERT INTO books (title, author, pdf\_path, cover\_image)

VALUES

('The Midnight Library', 'Matt Haig', 'midnight.pdf', 'midnight.jpg'),

('You Are a Badass', 'Jen Sincero', 'you.pdf', 'you.png'),

('The Silent Patient', 'Alex Michaelides', 'slient.pdf', 'slient.jpg');

-- Add books to user's library

INSERT INTO user\_library (user\_id, book\_id)

VALUES (1, 1), (1, 2), (1, 3);

SELECT b.title, b.author, b.pdf\_path, b.cover\_image

FROM user\_library ul

JOIN books b ON ul.book\_id = b.book\_id

WHERE ul.user\_id = 1;

USE booknooc;

CREATE TABLE mystery\_books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

image\_url VARCHAR(255),

description TEXT

);

INSERT INTO mystery\_books (title, image\_url, description) VALUES

('The Silent Witness', 'brunt.jpg', 'A case gone cold resurfaces after a decade.'),

('Beneath the Stairs', 'notebook.jpg', 'A chilling tale of a haunted house.'),

('Shadow in the Fog', 'secrets.jpg', 'A detective follows invisible trails through the mist.'),

('Whispers of the Past', 'guilty.jpg', 'History hides a dark truth.'),

('Locked Room Mystery', 'girl.jpg', 'A murder that defies logic and physics.');

SELECT \* FROM mystery\_books;

USE booknoob;

-- Users table

CREATE TABLE users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(100),

email VARCHAR(150)

);

-- Books table

CREATE TABLE books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255),

author VARCHAR(255),

cover\_image VARCHAR(255)

);

-- Orders table (purchases)

CREATE TABLE orders (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

book\_id INT,

order\_code VARCHAR(50),

purchase\_date DATE,

delivery\_status VARCHAR(100),

delivery\_date DATE,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (book\_id) REFERENCES books(book\_id)

);

-- Add a user

INSERT INTO users (username, email)

VALUES ('nalin', 'nalin@example.com');

-- Add books

INSERT INTO books (title, author, cover\_image)

VALUES

('The Midnight Library', 'Matt Haig', 'midnight.jpg'),

('The Silent Patient', 'Alex Michaelides', 'slient.jpg'),

('Steve Jobs', 'Walter Isaacson', 'jobs.jpg');

-- Add orders for user\_id = 1

INSERT INTO orders (user\_id, book\_id, order\_code, purchase\_date, delivery\_status, delivery\_date)

VALUES

(1, 1, '#BK234567', '2025-03-20', 'Delivered', '2025-03-24'),

(1, 2, '#BK239891', '2025-04-05', 'Out for Delivery', NULL),

(1, 3, '#BK245678', '2025-03-01', 'Delivered', '2025-03-04');

SELECT o.order\_code, b.title, b.author, b.cover\_image,

o.purchase\_date, o.delivery\_status, o.delivery\_date

FROM orders o

JOIN books b ON o.book\_id = b.book\_id

WHERE o.user\_id = 1;

CREATE TABLE IF NOT EXISTS about\_us (

id INT AUTO\_INCREMENT PRIMARY KEY,

heading VARCHAR(255) NOT NULL,

description TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Sample data

INSERT INTO about\_us (heading, description) VALUES

('Welcome to LibraryX', 'LibraryX is your one-stop platform for managing and exploring books. We aim to bring seamless access to literature and streamline book purchasing for all users.');

CREATE TABLE IF NOT EXISTS contact\_us (

message\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

subject VARCHAR(255),

message TEXT NOT NULL,

submitted\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Sample data

INSERT INTO contact\_us (name, email, subject, message) VALUES

('John Doe', 'john@example.com', 'Account Issue', 'I need help accessing my account.'),

('Jane Smith', 'jane@example.com', 'Book Inquiry', 'Do you have any books by Neil Gaiman?');

describe contact\_us;

CREATE TABLE Genre (

GenreID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL

);

describe genre;

CREATE TABLE BookGenre (

BookID INT,

GenreID INT,

PRIMARY KEY (BookID, GenreID),

FOREIGN KEY (BookID) REFERENCES Book(BookID),

FOREIGN KEY (GenreID) REFERENCES Genre(GenreID)

);

INSERT INTO Genre (Name, Description) VALUES

('Fiction', 'Narrative literature created from imagination.'),

('Non-Fiction', 'Factual books based on real events or people.'),

('Romantic', 'Love-centered stories with emotional themes.'),

('Biography', 'Accounts of a person\'s life written by someone else.'),

('Self-Help', 'Guides for personal growth, productivity, or mental well-being.'),

('Mystery', 'Stories involving investigations and suspense.'),

('Horror', 'Fiction intended to scare, disturb, or startle.'),

('Children', 'Books written specifically for young readers.'),

('Adventure', 'Action-packed stories involving risk and exploration.'),

('Science Fiction', 'Futuristic or technological imaginative stories.');

CREATE TABLE Publisher (

PublisherID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

Address TEXT

);

SELECT \* FROM Publisher;

-- 3. Author Table

CREATE TABLE Author (

AuthorID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

Bio TEXT

);

SELECT \* FROM Author;

CREATE TABLE Genre (

GenreID INTEGER PRIMARY KEY,

Name TEXT

);

CREATE TABLE Publisher (

PublisherID INTEGER PRIMARY KEY,

Name TEXT,

Address TEXT

);

CREATE TABLE Author (

AuthorID INTEGER PRIMARY KEY,

Name TEXT,

Bio TEXT

);

CREATE TABLE Customer (

CustomerID INTEGER PRIMARY KEY,

Name TEXT,

Email TEXT,

Phone TEXT,

Address TEXT

);

CREATE TABLE Book (

BookID INTEGER PRIMARY KEY,

Title TEXT,

GenreID INTEGER,

Price REAL,

PublisherID INTEGER,

Stock INTEGER,

FOREIGN KEY (GenreID) REFERENCES Genre(GenreID),

FOREIGN KEY (PublisherID) REFERENCES Publisher(PublisherID)

);

CREATE TABLE BookAuthor (

BookID INTEGER,

AuthorID INTEGER,

PRIMARY KEY (BookID, AuthorID),

FOREIGN KEY (BookID) REFERENCES Book(BookID),

FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID)

);

CREATE TABLE rder (

OrderID INTEGER PRIMARY KEY,

CustomerID INTEGER,

OrderDate TEXT,

TotalAmount REAL,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)

);

CREATE TABLE OrderDetails (

OrderDetailID INTEGER PRIMARY KEY,

OrderID INTEGER,

BookID INTEGER,

Quantity INTEGER,

FOREIGN KEY (OrderID) REFERENCES rder(OrderID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

CREATE TABLE Review (

ReviewID INTEGER PRIMARY KEY,

CustomerID INTEGER,

BookID INTEGER,

Rating INTEGER,

Comment TEXT,

ReviewDate TEXT,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

CREATE TABLE Wishlist (

WishlistID INTEGER PRIMARY KEY,

CustomerID INTEGER,

BookID INTEGER,

DateAdded TEXT,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

CREATE TABLE Employee (

EmployeeID INTEGER PRIMARY KEY,

Name TEXT,

Email TEXT,

Phone TEXT,

Position TEXT,

HireDate TEXT

);

CREATE TABLE Discount (

DiscountID INTEGER PRIMARY KEY,

BookID INTEGER,

DiscountPercentage REAL,

StartDate TEXT,

EndDate TEXT,

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

CREATE TABLE Shipment (

ShipmentID INTEGER PRIMARY KEY,

OrderID INTEGER,

ShipmentDate TEXT,

DeliveryDate TEXT,

Status TEXT,

TrackingNumber TEXT,

FOREIGN KEY (OrderID) REFERENCES rder(OrderID)

);

CREATE TABLE Payment (

PaymentID INTEGER PRIMARY KEY,

OrderID INTEGER,

PaymentDate TEXT,

PaymentMethod TEXT,

Amount REAL,

FOREIGN KEY (OrderID) REFERENCES rder(OrderID)

);

CREATE TABLE ReturnRequest (

ReturnID INTEGER PRIMARY KEY,

OrderID INTEGER,

BookID INTEGER,

RequestDate TEXT,

Reason TEXT,

Status TEXT,

FOREIGN KEY (OrderID) REFERENCES rder(OrderID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

INSERT INTO Genre VALUES (1, 'Fiction'), (2, 'Non-fiction'), (3, 'Romantic'), (4, 'Biography'), (5, 'Self-help'), (6, 'Mystery'), (7, 'Horror'), (8, 'Children'), (9, 'Science'), (10, 'Fantasy');

INSERT INTO Publisher VALUES (1, 'Penguin Random House', 'New York'), (2, 'HarperCollins', 'London');

INSERT INTO Author VALUES (1, 'John Doe', 'Bestselling fiction author'), (2, 'Jane Smith', 'Expert in psychology');

INSERT INTO Customer VALUES (1, 'Alice Johnson', 'alice@example.com', '1234567890', '123 Maple Street'),

(2, 'Bob Martin', 'bob@example.com', '0987654321', '456 Oak Avenue');

INSERT INTO Book VALUES (1, 'The Lost Island', 1, 350.00, 1, 100),

(2, 'Mind Matters', 5, 499.99, 2, 50);

INSERT INTO BookAuthor VALUES (1, 1), (2, 2);

INSERT INTO rder VALUES (1, 1, '2025-04-01', 849.99);

INSERT INTO OrderDetails VALUES (1, 1, 1, 1), (2, 1, 2, 1);

INSERT INTO Review VALUES (1, 1, 1, 5, 'Loved it!', '2025-04-02');

INSERT INTO Wishlist VALUES (1, 2, 1, '2025-04-05');

INSERT INTO Employee VALUES (1, 'Michael Scott', 'michael@bookstore.com', '9999999999', 'Manager', '2023-01-10');

INSERT INTO Discount VALUES (1, 2, 10.0, '2025-04-01', '2025-04-30');

INSERT INTO Shipment VALUES (1, 1, '2025-04-02', '2025-04-06', 'Delivered', 'TRK123456');

INSERT INTO Payment VALUES (1, 1, '2025-04-01', 'Credit Card', 849.99);

INSERT INTO ReturnRequest VALUES (1, 1, 2, '2025-04-07', 'Damaged copy', 'Pending');

tables = ["Genre", "Publisher", "Author", "Customer", "Book", "BookAuthor", "Order", "OrderDetails",

"Review", "Wishlist", "Employee", "Discount", "Shipment", "Payment", "ReturnRequest"

]

results = SELECT \* FROM table

conn.close()

results.keys()

-- 4. Customer Table

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15),

Address TEXT

);

SELECT \* FROM Customer;

-- 5. Book Table

CREATE TABLE Book (

BookID INT PRIMARY KEY AUTO\_INCREMENT,

Title VARCHAR(200) NOT NULL,

GenreID INT,

Price DECIMAL(10, 2),

PublisherID INT,

Stock INT,

FOREIGN KEY (GenreID) REFERENCES Genre(GenreID),

FOREIGN KEY (PublisherID) REFERENCES Publisher(PublisherID)

);

SELECT \* FROM Book;

-- 6. BookAuthor Table

CREATE TABLE BookAuthor (

BookID INT,

AuthorID INT,

PRIMARY KEY (BookID, AuthorID),

FOREIGN KEY (BookID) REFERENCES Book(BookID),

FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID)

);

SELECT \* FROM BookAuthor;

-- 7. Order Table

CREATE TABLE `Order` (

OrderID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10, 2),

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)

);

SELECT \* FROM `Order`;

-- 8. OrderDetails Table

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

BookID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

SELECT \* FROM OrderDetails;

-- 9. Review Table

CREATE TABLE Review (

ReviewID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

BookID INT,

Rating INT CHECK (Rating BETWEEN 1 AND 5),

Comment TEXT,

ReviewDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

SELECT \* FROM Review;

-- 10. Wishlist Table

CREATE TABLE Wishlist (

WishlistID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

BookID INT,

DateAdded DATE,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

SELECT \* FROM Wishlist;

-- 11. Employee Table

CREATE TABLE Employee (

EmployeeID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100),

Email VARCHAR(100),

Phone VARCHAR(15),

Position VARCHAR(50),

HireDate DATE

);

SELECT \* FROM Employee;

-- 12. Discount Table

CREATE TABLE Discount (

DiscountID INT PRIMARY KEY AUTO\_INCREMENT,

BookID INT,

DiscountPercentage DECIMAL(5,2),

StartDate DATE,

EndDate DATE,

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

SELECT \* FROM Discount;

-- 13. Shipment Table

CREATE TABLE Shipment (

ShipmentID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

ShipmentDate DATE,

DeliveryDate DATE,

Status VARCHAR(50),

TrackingNumber VARCHAR(100),

FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID)

);

SELECT \* FROM Shipment;

-- 14. Payment Table

CREATE TABLE Payment (

PaymentID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

PaymentDate DATE,

PaymentMethod VARCHAR(50),

Amount DECIMAL(10,2),

FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID)

);

SELECT \* FROM Payment;

-- 15. ReturnRequest Table

CREATE TABLE ReturnRequest (

ReturnID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

BookID INT,

RequestDate DATE,

Reason TEXT,

Status VARCHAR(50),

FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

SELECT \* FROM ReturnRequest;

CREATE TABLE Wishlist (

WishlistID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

BookID INT,

DateAdded DATE,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

describe Wishlist;

CREATE TABLE Employee (

EmployeeID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100),

Email VARCHAR(100),

Phone VARCHAR(15),

Position VARCHAR(50),

HireDate DATE

);

describe Employee;

-- Sample Queries -- 1. List all books with their genre and publisher SELECT b.Title, g.Name AS Genre, p.Name AS Publisher FROM Book b JOIN Genre g ON b.GenreID = g.GenreID JOIN Publisher p ON b.PublisherID = p.PublisherID;

-- Find top 5 books with highest rating SELECT b.Title, AVG(r.Rating) AS AverageRating FROM Review r JOIN Book b ON r.BookID = b.BookID GROUP BY r.BookID ORDER BY AverageRating DESC LIMIT 5;

-- Show customer order history SELECT c.Name, o.OrderID, o.OrderDate, o.TotalAmount FROM Customer c JOIN Order o ON c.CustomerID = o.CustomerID;

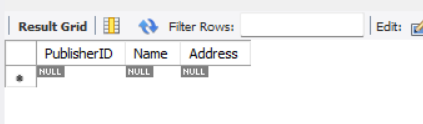
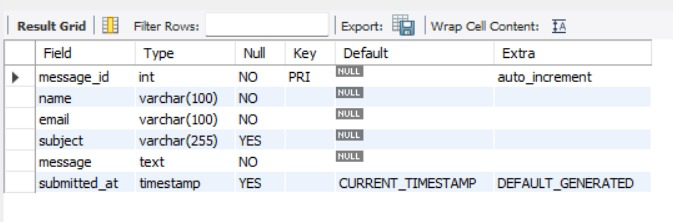
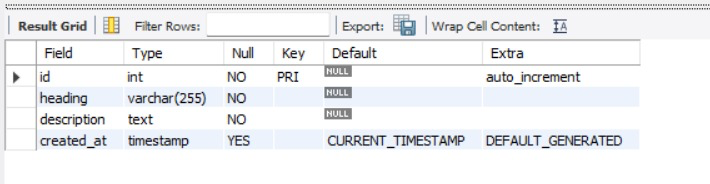
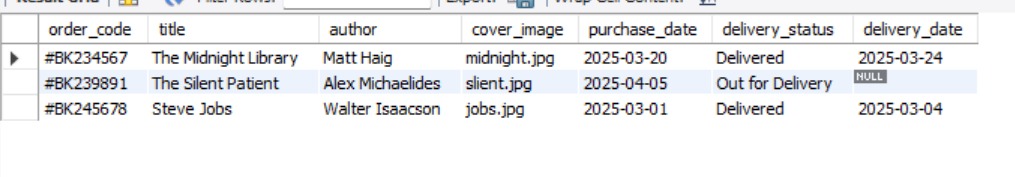
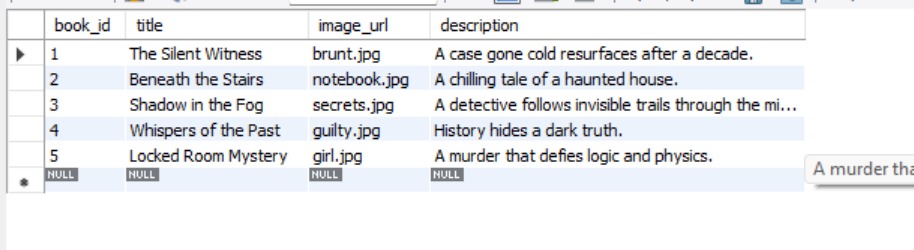
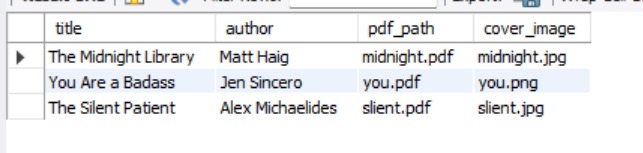
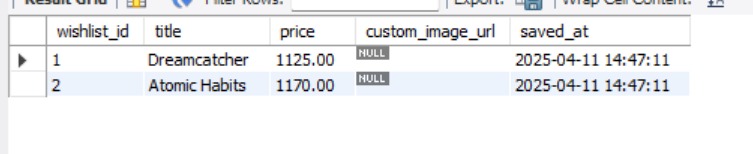
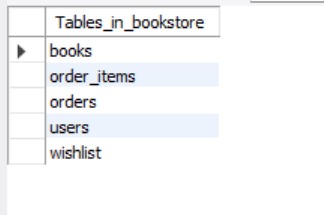
-- Books currently on discount SELECT b.Title, d.DiscountPercentage FROM Book b JOIN Discount d ON b.BookID = d.BookID WHERE CURRENT\_DATE BETWEEN d.StartDate AND d.EndDate;

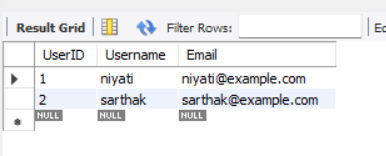
-- Get wishlist books for a specific customer SELECT c.Name, b.Title FROM Wishlist w JOIN Customer c ON w.CustomerID = c.CustomerID JOIN Book b ON w.BookID = b.BookID WHERE c.CustomerID = 1;

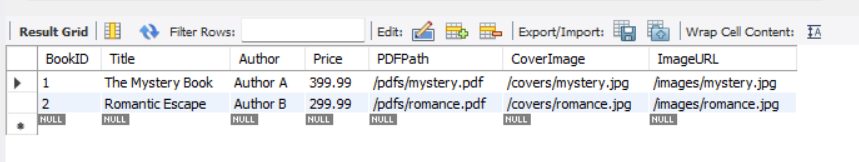
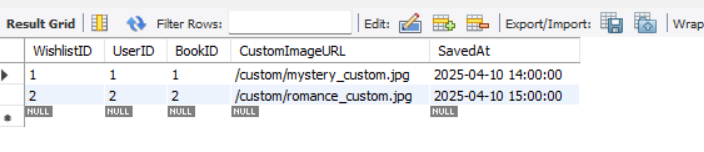
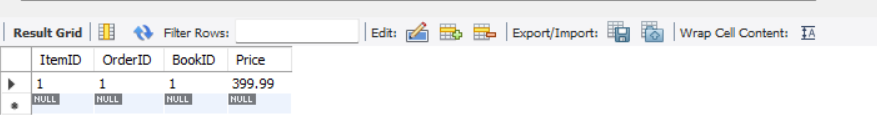
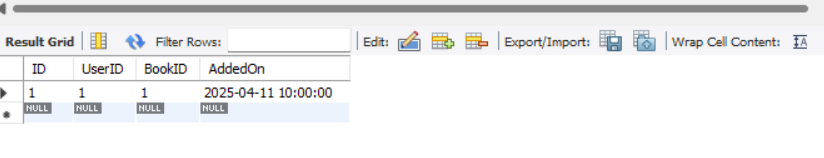
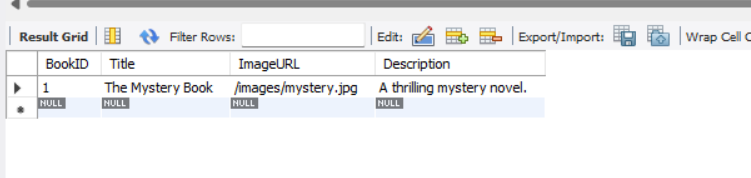
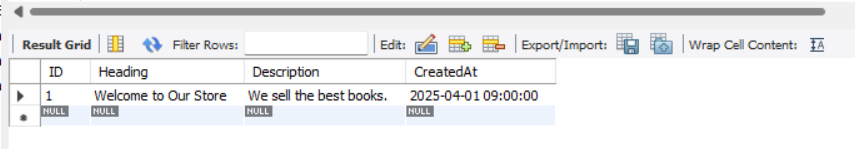
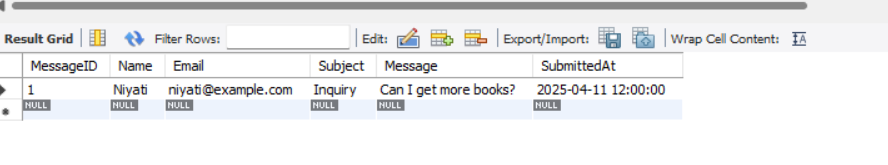
-- Total revenue from orders SELECT SUM(TotalAmount) AS TotalRevenue FROM Order;

-- List all shipments with delivery status SELECT s.ShipmentID, s.Status, o.OrderID, c.Name FROM Shipment s JOIN Order o ON s.OrderID = o.OrderID JOIN Customer c ON o.CustomerID = c.CustomerID;

**OUTPUT:**

**       **



**Relational schema:**

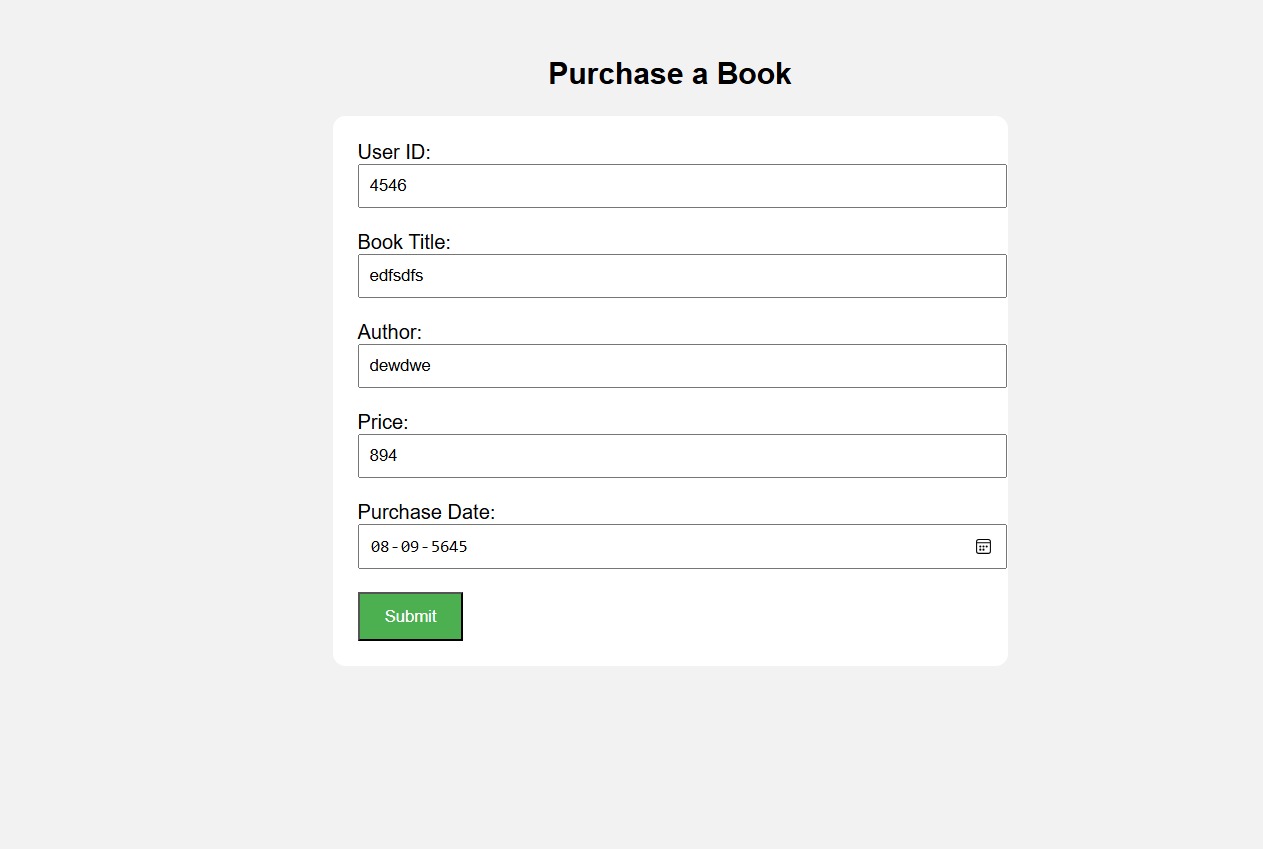
1. Genre(GenreID [PK], Name)
2. Publisher(PublisherID [PK], Name, Address)
3. Author(AuthorID [PK], Name, Bio)
4. Customer(CustomerID [PK], Name, Email, Phone, Address)
5. Book(BookID [PK], Title, GenreID [FK], Price, PublisherID [FK], Stock)
   1. GenreID → Genre(GenreID)
   2. PublisherID → Publisher(PublisherID)
6. BookAuthor(BookID [PK, FK], AuthorID [PK, FK])
   1. Composite Primary Key: (BookID, AuthorID)
   2. BookID → Book(BookID)
   3. AuthorID → Author(AuthorID)
7. Order(OrderID [PK], CustomerID [FK], OrderDate, TotalAmount)
   1. CustomerID → Customer(CustomerID)
8. OrderDetails(OrderDetailID [PK], OrderID [FK], BookID [FK], Quantity)
   1. OrderID → Order(OrderID)
   2. BookID → Book(BookID)
9. Review(ReviewID [PK], CustomerID [FK], BookID [FK], Rating, Comment, ReviewDate)
   1. CustomerID → Customer(CustomerID)
   2. BookID → Book(BookID)
10. Wishlist(WishlistID [PK], CustomerID [FK], BookID [FK], DateAdded)
    1. CustomerID → Customer(CustomerID)
    2. BookID → Book(BookID)
11. Employee(EmployeeID [PK], Name, Email, Phone, Position, HireDate)
12. Discount(DiscountID [PK], BookID [FK], DiscountPercentage, StartDate, EndDate)
    1. BookID → Book(BookID)
13. Shipment(ShipmentID [PK], OrderID [FK], ShipmentDate, DeliveryDate, Status, TrackingNumber)
    1. OrderID → Order(OrderID)
14. Payment(PaymentID [PK], OrderID [FK], PaymentDate, PaymentMethod, Amount)
    1. OrderID → Order(OrderID)
15. ReturnRequest(ReturnID [PK], OrderID [FK], BookID [FK], RequestDate, Reason, Status)
    1. OrderID → Order(OrderID)
    2. BookID → Book(BookID)
16. Books(BookID [PK], Title, Price, ImageURL)
17. Users(UserID [PK], Username, Email)
18. Wishlist(WishlistID [PK], UserID [FK], BookID [FK], SavedAt)
19. Orders(OrderID [PK], UserID [FK], TotalAmount, PaymentMethod, CouponCode, GiftcardCode, OrderedAt)
20. OrderItems(ItemID [PK], OrderID [FK], BookID [FK], Price)
21. Books(BookID [PK], Title, Author, PDFPath, CoverImage)
22. UserLibrary(ID [PK], UserID [FK], BookID [FK], AddedOn)
23. Wishlist(WishlistID [PK], UserID [FK], BookID [FK], CustomImageURL, SavedAt)
24. MysteryBooks(BookID [PK], Title, ImageURL, Description)  
    AboutUs(ID [PK], Heading, Description, CreatedAt)  
    ContactUs(MessageID [PK], Name, Email, Subject, Message, SubmittedAt)
25. Genre Genre(GenreID [PK], Name)
26. Publisher Publisher(PublisherID [PK], Name, Address)
27. Author Author(AuthorID [PK], Name, Bio)
28. Customer Customer(CustomerID [PK], Name, Email, Phone, Address)
29. Book Book(BookID [PK], Title, GenreID [FK], Price, PublisherID [FK], Stock)
    1. GenreID → Genre(GenreID)
    2. PublisherID → Publisher(PublisherID)
30. BookAuthor BookAuthor(BookID [PK, FK], AuthorID [PK, FK])
    1. Composite Primary Key: (BookID, AuthorID)
    2. BookID → Book(BookID)
    3. AuthorID → Author(AuthorID)
31. Order Order(OrderID [PK], CustomerID [FK], OrderDate, TotalAmount)
    1. CustomerID → Customer(CustomerID)
32. OrderDetails OrderDetails(OrderDetailID [PK], OrderID [FK], BookID [FK], Quantity)
    1. OrderID → Order(OrderID)
    2. BookID → Book(BookID)
33. Review Review(ReviewID [PK], CustomerID [FK], BookID [FK], Rating, Comment, ReviewDate)
    1. CustomerID → Customer(CustomerID)
    2. BookID → Book(BookID)
34. Wishlist Wishlist(WishlistID [PK], CustomerID [FK], BookID [FK], DateAdded)
    1. CustomerID → Customer(CustomerID)
    2. BookID → Book(BookID)
35. Employee Employee(EmployeeID [PK], Name, Email, Phone, Position, HireDate)
36. Discount Discount(DiscountID [PK], BookID [FK], DiscountPercentage, StartDate, EndDate)
    1. BookID → Book(BookID)
37. Shipment Shipment(ShipmentID [PK], OrderID [FK], ShipmentDate, DeliveryDate, Status, TrackingNumber)
    1. OrderID → Order(OrderID)
38. Payment Payment(PaymentID [PK], OrderID [FK], PaymentDate, PaymentMethod, Amount)
    1. OrderID → Order(OrderID)
39. ReturnRequest ReturnRequest(ReturnID [PK], OrderID [FK], BookID [FK], RequestDate, Reason, Status)
    1. OrderID → Order(OrderID)
    2. BookID → Book(BookID)

**VI. Project demonstration**

**Tools/Software Used:**

* MySQL
* PHPMyAdmin
* VS Code

**GUI Screenshot and Description:**



-- Create the database if it doesn't exist

CREATE DATABASE IF NOT EXISTS library\_db;

USE library\_db;

-- Create users table

CREATE TABLE IF NOT EXISTS users (

user\_id INT PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100)

);

-- Insert sample users

INSERT INTO users (user\_id, name, email) VALUES

(1, 'Alice Johnson', 'alice@example.com'),

(2, 'Bob Smith', 'bob@example.com'),

(3, 'Charlie Brown', 'charlie@example.com');

describe users;

-- Create books table

CREATE TABLE IF NOT EXISTS books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_title VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

price DECIMAL(10,2) NOT NULL

);

-- Insert sample books

INSERT INTO books (book\_title, author, price) VALUES

('The Great Gatsby', 'F. Scott Fitzgerald', 399.99),

('To Kill a Mockingbird', 'Harper Lee', 299.50),

('1984', 'George Orwell', 349.00),

('The Catcher in the Rye', 'J.D. Salinger', 289.00);

describe books;

-- Create purchases table

CREATE TABLE IF NOT EXISTS purchases (

purchase\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

book\_id INT NOT NULL,

purchase\_date DATE NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (book\_id) REFERENCES books(book\_id)

);

-- Insert sample purchases

INSERT INTO purchases (user\_id, book\_id, purchase\_date) VALUES

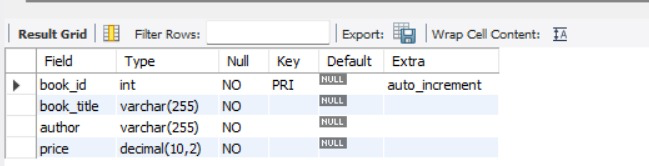
(1, 1, '2024-12-10'),

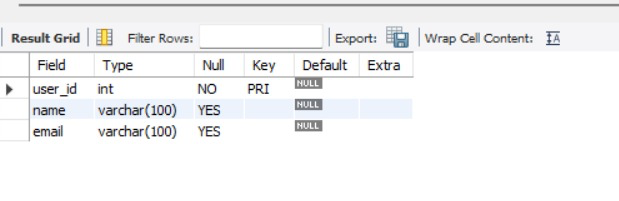
(2, 2, '2025-01-15'),

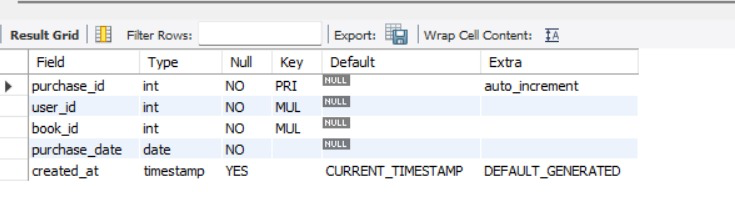
(3, 3, '2025-02-20'),

(1, 4, '2025-03-05');

describe purchases;







**Self -Learning beyond classroom**

:

* Learned ER modeling tools like Draw.io
* Understood advanced SQL features like nested queries and joins
* Gained experience with XAMPP and local server hosting
* Explored database design principles

**Learning from the Project**

* Real-life application of DBMS concepts
* Improved skills in designing normalized databases
* Gained practical SQL query experience
* Understood the importance of structured data management

**Challenges Faced**

* Designing many-to-many relationships
* Ensuring database normalization
* Debugging syntax errors in SQL

**Conclusion**

This project helped solidify understanding of database management systems through a real-world use case. We learned to design, implement, and query a relational database effectively. The hands-on experience enhanced our confidence and skill-set in managing database-driven applications.