**Online Bookstore Inventory Management System**

**Introduction**

The **Online Bookstore Inventory Management System** is a digital platform developed to automate and manage various tasks related to running an online bookstore. This system ensures smooth operation by organizing data and processes efficiently.

The system ensures:

* Efficient book and stock management
* Order processing and customer tracking
* Categorization of books
* Admin activity tracking

**Key Functions of the System:**

* **Book Management**
  + Add, update, and remove book listings
  + Track book details like title, author, genre, price, and stock
* **Category Management**
  + Organize books into categories (e.g., fiction, academic, self-help)
  + Allow admins to create and manage categories easily
* **Customer Management**
  + Store customer information (name, contact, address, registration date)
  + Maintain a record of returning and new customers
* **Order and Purchase Management**
  + Customers can place orders for one or more books
  + System calculates total amount, tracks payment and delivery status
* **Inventory Tracking**
  + Automatically update stock levels after each order
  + Prevent overselling and notify low-stock items
* **Admin Controls**
  + Admins can manage books, categories, and view overall system activities
  + Secure control over who can create or modify entries
* **Customer Reviews & Ratings**
  + Customers can give feedback and rate books
  + Helps other buyers in decision-making and improves service

**System Benefits:**

* Reduces manual work and paperwork
* Minimizes human errors in orders and inventory
* Provides real-time access to data (books, orders, customers)
* Improves customer experience with a smooth and organized process
* Easy to use, even for users with basic technical skills

**Tables and Entities:**

It includes 7 main tables: Admins, Categories, Books, Customers, Orders, Order\_Items, and Reviews. Each table handles specific tasks such as book records, customer data, orders, and feedback.

**1. Admins**

admin\_id, name, email, phone, role, created\_at

*Primary Key:* admin\_id

*Foreign Key:* None

**2. Categories**

category\_id, name, description, created\_by, created\_at, status

*Primary Key:* category\_id

*Foreign Key: c*reated\_by → Admins(admin\_id)

**3. Books**

Book\_id, Title, Author, Genre, Price, Stock, Category\_id

*Primary Key:* Book\_id

*Foreign Key:* Category\_id → Categories(Category\_id)

**4. Customers**

customer\_id, name, email, phone, address, registration\_date

*Primary Key:* customer\_id

*Foreign Key:* none

**5. Orders**

order\_id, customer\_id, order\_date, total\_amount, payment\_status, delivery\_status

Primary Key: order\_id

Foreign Key: customer\_id → Customers(customer\_id)

**6. Order\_Items**

item\_id, order\_id, book\_id, quantity, price, subtotal

Primary Key: item\_id

Foreign Keys: order\_id → Orders(order\_id)

: book\_id → Books(book\_id)

**7. Reviews**

review\_id, book\_id, customer\_id, rating, comment, review\_date

Primary Key: review\_id

Foreign Keys: book\_id → Books(book\_id)

: customer\_id → Customers(customer\_id)

Relationships Among Tables:

**1. Customers — Places —> Orders**

Cardinality: Customer (M) — (M) Order

**2. Order\_Items — Contains —> Books**

Cardinality: Order\_Items (M) — (M) Books

**3. Customer — Writes —> Review**

Cardinality: Customer (1) — (M) Review

**4. Reviews — Of —> Books**

Cardinality: Book (M) — (M) Review

**5. Admins — Create —> Category**

Cardinality: Admin (1) — (M) Category

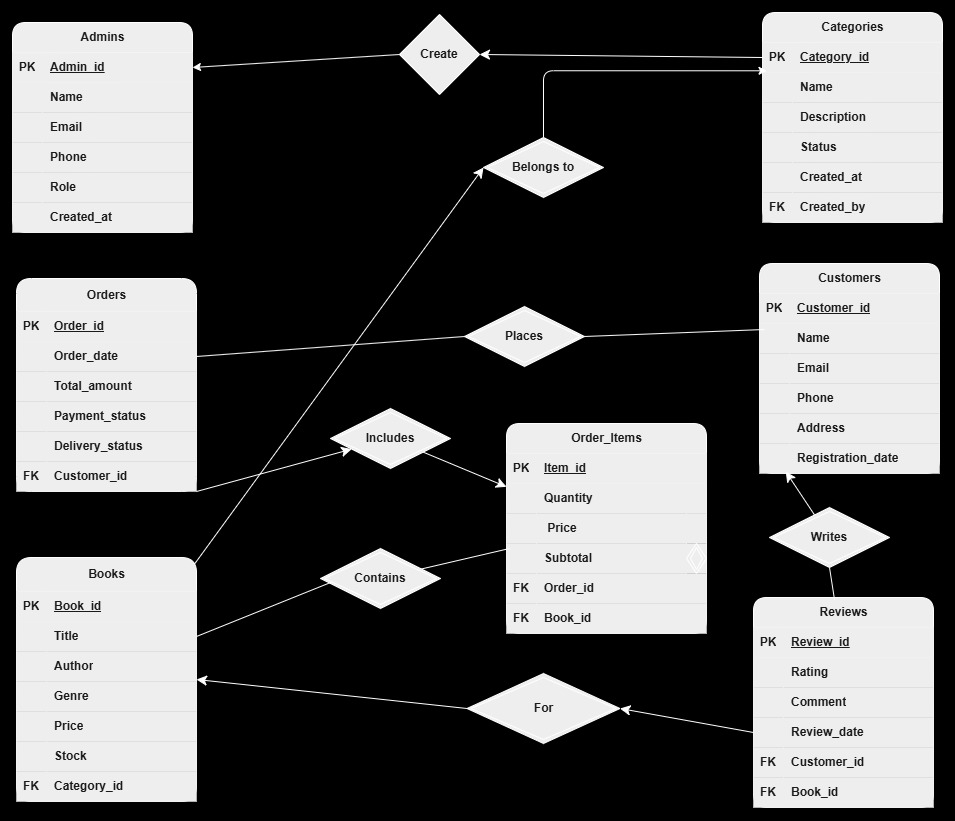
**6. Books — Belong\_to —> Category**

Cardinality: Category (1) — (M) Books

**7. Orders — Include —> Order\_Items**

Cardinality: Orders (M) — (M) Order\_Items

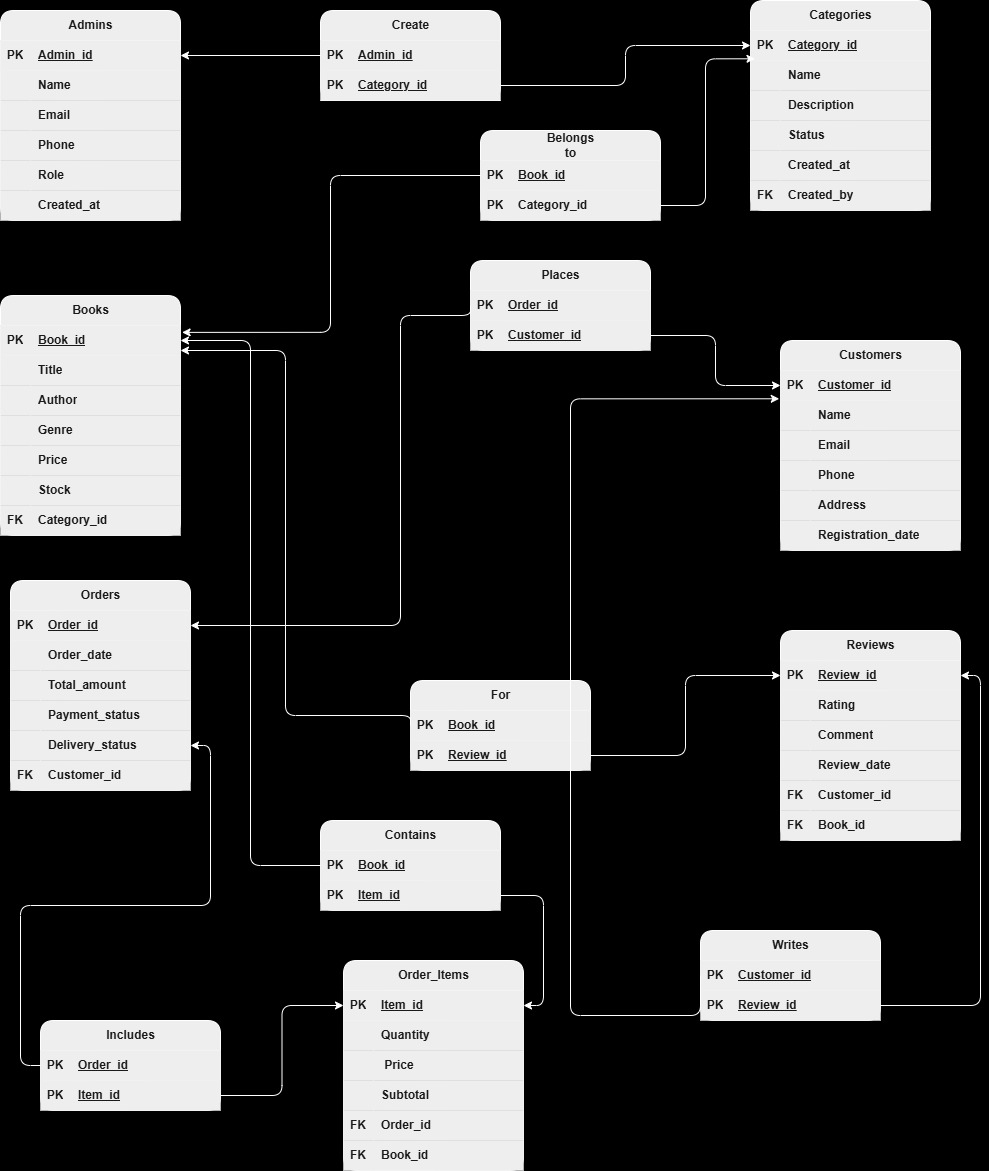
**ER Diagram:**



For clear understanding:

* + 1. [https://app.diagrams.net/#G1qXH6PQWHstDJHyRWhLUKhlL9g9o5K2x1#%7B%22pageId%22%3A%22bPgeqzEszBXqpjA1GqrR%22%7D](https://app.diagrams.net/#G1qXH6PQWHstDJHyRWhLUKhlL9g9o5K2x1)
    2. <https://drive.google.com/file/d/1qXH6PQWHstDJHyRWhLUKhlL9g9o5K2x1/view?usp=drive_link>

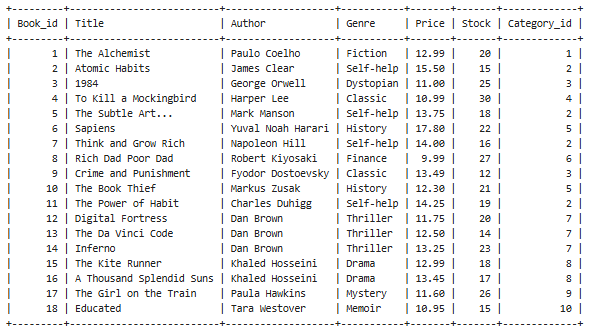
**Schema Diagram:**



**Queries:**

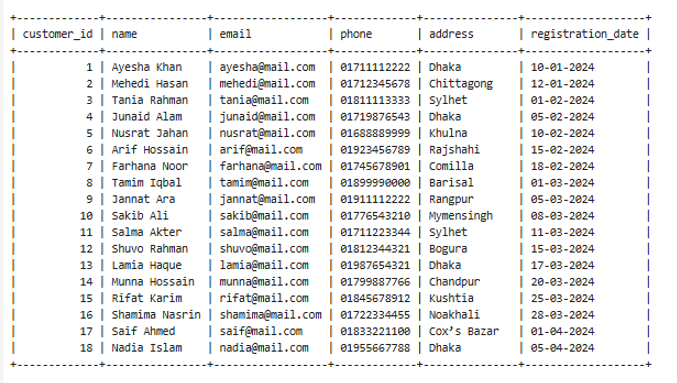
1. (**Display all columns and all records from the Books table.**)

SELECT \* FROM Books



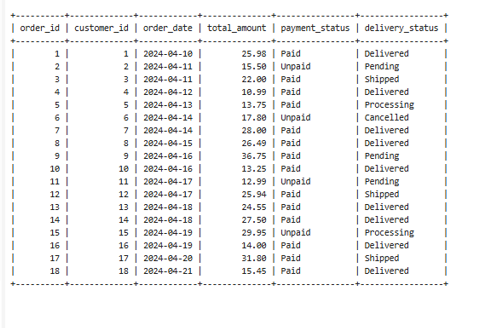
1. (**Display all columns and all records from the** Customers **table.**)

SELECT \* FROM Customers



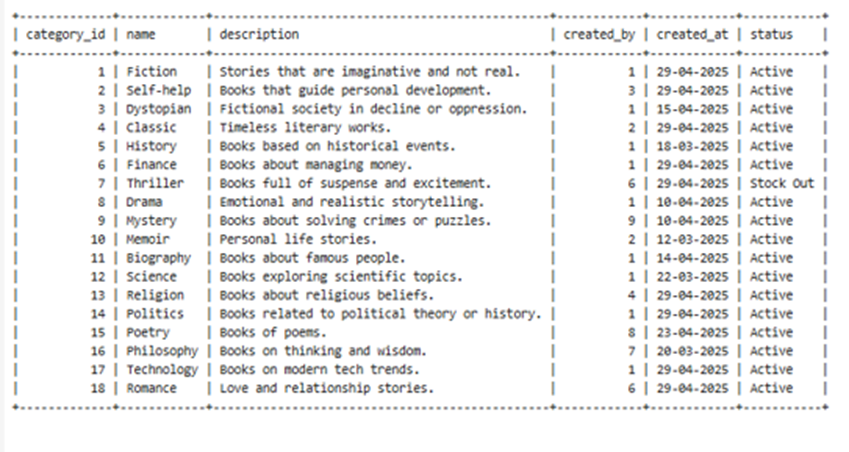
1. (**Display all columns and all records from the** Orders **table.**)

SELECT \* FROM Orders



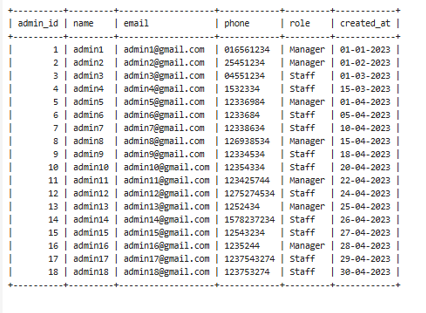
1. (**Display all columns and all records from the** Categories **table.**)

SELECT \* FROM Categories



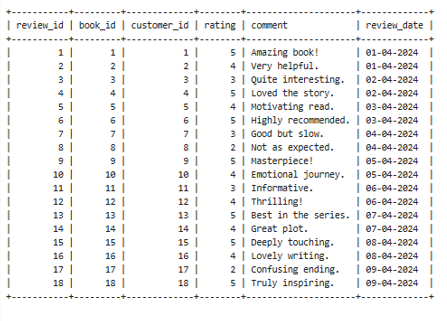
1. (**Display all columns and all records from the Admins table.**)

SELECT \* FROM Admins



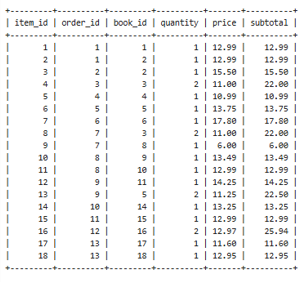
1. **(Display all columns and all records from the Reviews table.)**

SELECT \* FROM Reviews



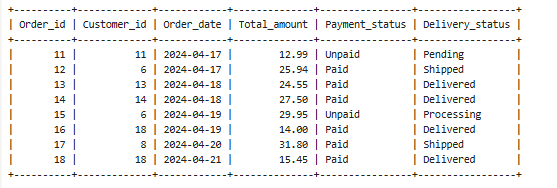
1. (**Display all columns and all records from the Order\_Items table.**)

SELECT \* FROM Order\_Items



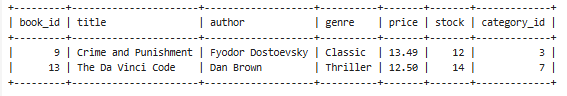
1. **(Retrieve all orders placed between April 17, 2024, and April 30, 2024.)**

SELECT \* FROM Orders WHERE order\_date BETWEEN '2024-04-17' AND '2024-04-30';



1. **(Fetche all books with stock less than 15 units.)**

SELECT \* FROM Books WHERE stock < 15;



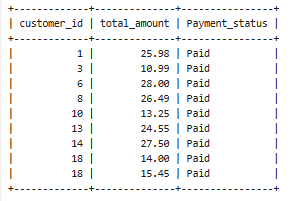
1. **(Retrieve all admin records where the role is 'Manager’)**

SELECT \* FROM Admins WHERE role = 'Manager';



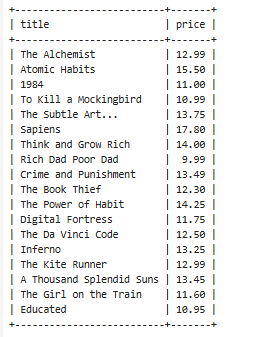
1. **(Fetche customer IDs, total amounts, and payment statuses for all orders that have been delivered.)**

SELECT customer\_id,total\_amount,Payment\_status FROM Orders WHERE delivery\_status = 'Delivered';



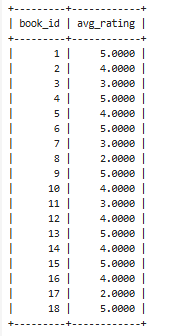
1. **(Retrieve the titles and prices of all books in the inventory.)**

SELECT title, price FROM Books;



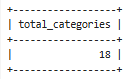
1. **(Calculate the average rating for each book based on customer reviews)**

SELECT book\_id, AVG(rating) AS avg\_rating FROM Reviews GROUP BY book\_id;



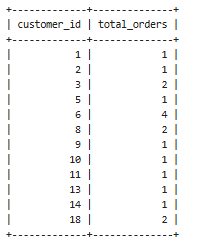
1. **(Count the total number of categories in the system.)**

SELECT COUNT(\*) AS total\_categories FROM Categories;



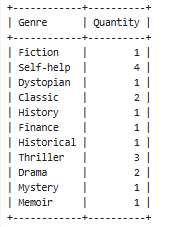
1. **(Show the total number of orders placed by each customer.)**

SELECT customer\_id, COUNT(\*) AS total\_orders FROM Orders GROUP BY customer\_id;



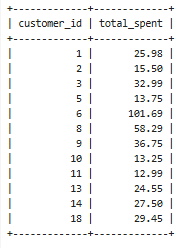
1. **(Count the number of books available in each genre)**

SELECT Genre , count(Genre) as Quantity from Books GROUP BY(Genre);



1. **(Calculate the total amount spent by each customer across all their orders.)**

SELECT customer\_id, SUM(total\_amount) AS total\_spent FROM Orders GROUP BY customer\_id ;



1. **(Add a new column preferred\_genre to the Customers table and updates each customer with their preferred book genre, then displays their name, district, registration date, and preferred genre.)**

ALTER TABLE Customers

ADD column preferred\_genre VARCHAR(50);

UPDATE Customers SET preferred\_genre = 'Fiction' WHERE customer\_id = 1;

UPDATE Customers SET preferred\_genre = 'Self-help' WHERE customer\_id = 2;

UPDATE Customers SET preferred\_genre = 'Mystery' WHERE customer\_id = 3;

UPDATE Customers SET preferred\_genre = 'Drama' WHERE customer\_id = 4;

UPDATE Customers SET preferred\_genre = 'Romance' WHERE customer\_id = 5;

UPDATE Customers SET preferred\_genre = 'Science' WHERE customer\_id = 6;

UPDATE Customers SET preferred\_genre = 'Thriller' WHERE customer\_id = 7;

UPDATE Customers SET preferred\_genre = 'History' WHERE customer\_id = 8;

UPDATE Customers SET preferred\_genre = 'Classic' WHERE customer\_id = 9;

UPDATE Customers SET preferred\_genre = 'Poetry' WHERE customer\_id = 10;

UPDATE Customers SET preferred\_genre = 'Technology' WHERE customer\_id = 11;

UPDATE Customers SET preferred\_genre = 'Biography' WHERE customer\_id = 12;

UPDATE Customers SET preferred\_genre = 'Philosophy' WHERE customer\_id = 13;

UPDATE Customers SET preferred\_genre = 'Dystopian' WHERE customer\_id = 14;

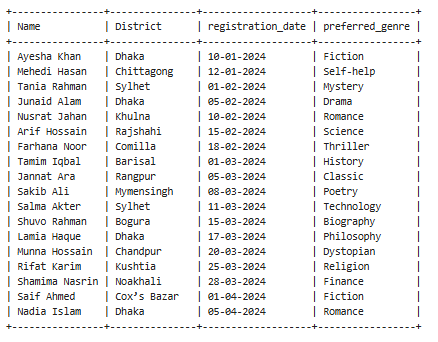
UPDATE Customers SET preferred\_genre = 'Religion' WHERE customer\_id = 15;

UPDATE Customers SET preferred\_genre = 'Finance' WHERE customer\_id = 16;

UPDATE Customers SET preferred\_genre = 'Fiction' WHERE customer\_id = 17;

UPDATE Customers SET preferred\_genre = 'Romance' WHERE customer\_id = 18;

SELECT Name, Address as District, registration\_date, preferred\_genre FROM Customers;



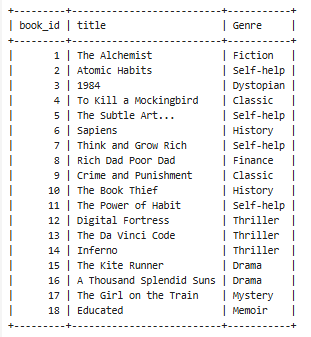
1. **(Retrieve all customers who have placed at least one order.)**

SELECT \* FROM Customers WHERE customer\_id IN (SELECT customer\_id FROM Orders);



1. **(Fetche details of books that have received at least one review.)**

SELECT book\_id,title,Genre FROM Books WHERE book\_id IN (SELECT book\_id FROM Reviews);



1. **(Retrieve all books priced below the average price of all books.)**

SELECT \* FROM Books WHERE price < (SELECT AVG(price) FROM Books);



1. **(Display the top 8 highest-rated books based on average review ratings, in descending order.)**

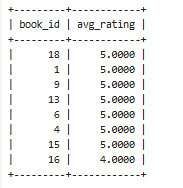
SELECT book\_id, AVG(rating) AS avg\_rating

FROM Reviews

GROUP BY book\_id

ORDER BY avg\_rating DESC

LIMIT 8;



**Conclusion:**  
This report presents a comprehensive overview of the bookstore database through various SQL queries, providing insights into book inventory, customer preferences, sales performance, and administrative roles. The data retrieved supports efficient decision-making by identifying popular genres, top-rated books, customer purchase behavior, and inventory needs. These findings can help improve customer satisfaction, optimize stock management, and enhance overall business operations.