**Online Bookstore Management System**

*Project Report with ER Model Reference*

## **1. Introduction : The Digital Bookshelf**

## Think about your favourite local bookstore. Now, imagine the owner trying to keep track of every single book, every customer's order, and every sale using just spreadsheets or, even worse, a paper ledger. It would be a nightmare. A single typo could mean a book is listed as in-stock when it's not, leading to a disappointed customer. That's the problem we're solving. We're moving away from those clunky, old-school methods that are slow and full of potential mistakes. This project is about building a smart, centralized system that handles all of that automatically. It's a single place to manage everything—customer details, the book inventory, and all the transactions—smoothly and securely. Essentially, we're creating a reliable digital backbone for the bookstore, so the owners can spend less time fighting with paperwork and more time connecting people with books they'll love.

## **2. Problem Overview**

* **Manual Processing Delays** -> Slow retrieval and updating of information.
* **Data Redundancy & Inconsistency** -> Storing the same information (e.g., customer details) in multiple locations, leading to inconsistencies.
* **Lack of Data Integrity** -> High risk of human error during manual entry, resulting in inaccurate inventory counts, incorrect order details, and unreliable reporting.
* **Inefficient operations** -> Unprotected or poorly managed records pose significant threats.

A well-structured system eliminates these problems by providing a centralized, relational database with properly defined entities, relationships, and constraints.

## **3. Scope of the Project**

Designing a Relational Database

* **Comprehensive Data Storage** - Store all details of customers ,books, orders, employees, and suppliers.
* **Optimized Relationships** - Represent complex relationships between entities in a logical and optimized structure
* **Fast Retrieval & Updates** - Enable quick access and modification of records via structured queries.
* **Scalability & Security** - Support future growth and ensure data security with access control.

## **4. Objectives**

**1. Centralized Data Management** - Unified system, eliminating duplication.

**2. Data Retrieval** - Fast and accurate information access.

**3. Relationship Mapping -** Handle one-to-many and many-to-many relationships.

**4. Security & Access Control** - Protect sensitive information.

**5. Scalability & Maintainability** - Handle future growth and easy updates.

**6. Reporting & Analytics** - Generate comprehensive reports.

**7. Minimizing Redundancy** - Normalization for data consistency.

## 

## **5. Significance of the Project**

* **Reduced Operational Delays :** Streamlined processes for faster service.
* **Improved Accuracy :** Reliable and consistent records.

* **Enhanced Customer Satisfaction :** Quicker and more efficient service.
* **Strengthened Security :** Protection of sensitive financial data.

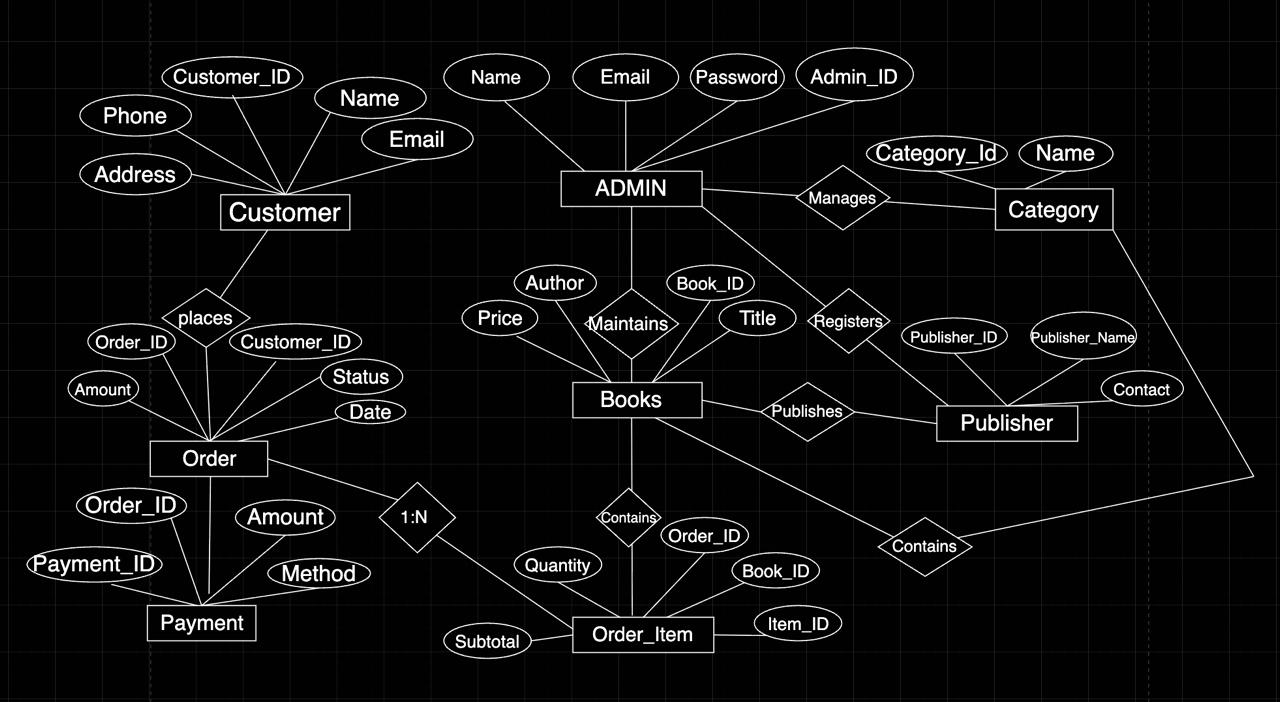
**6. Main Entities and Attributes**

|  |  |
| --- | --- |
| **Entity** | **Attributes** |
| Admin | \*AdminID (Primary Key)  \* Name  \* Email  \* Password |
| Customer | \* CustomerID (Primary Key)  \* Name  \* Email  \* Phone  \* Address |
| Order | \* OrderID (Primary Key)  \* CustomerID (Foreign Key)  \* OrderDate  \* TotalAmount  \* Status |
| Payment | \* PaymentID (Primary Key)  \* OrderID (Foreign Key)  \* PaymentDate  \* Amount  \* PaymentMethod |
| OrderItem | \* OrderItemID (Primary Key)  \* OrderID (Foreign Key)  \* BookID (Foreign Key)  \* Quantity  \* Subtotal |
| Book | \* BookID (Primary Key)  \* Title  \* Author  \* ISBN  \* Price  \* StockQuantity  \* PublisherID (Foreign Key)  \* CategoryID (Foreign Key) |
| Category | \* CategoryID (Primary Key)  \* CategoryName |
| Publisher | \* PublisherID (Primary Key)  \* PublisherName  \* Contact |
|  |  |
|  |  |
|  |  |

**Relationships**:

* Customer3Order: One-to-Many
* Book3Order\_Item: One-to-Many
* Order3Order\_Item: One-to-Many
* Book3Supplier: Many-to-Many (via a separate linking table if needed)

7. **ER MODEL :**



## 

**ER to Relational Model (Schema):**

Customer(Customer\_ID [PK],Name, Address, Phone, Email,Date\_of\_Birth)

Book(Book\_ID [PK], Title, Author, Genre, Price, Stock\_Quantity)

Order(Order\_ID [PK], Customer\_ID [FK], Order\_Date, Total\_Amount, Status)

Order\_Item(Order\_ID [FK], Book\_ID [FK], Quantity, Subtotal) 3 Composite PK

Employee(Employee\_ID [PK], Name, Designation, Salary)

Supplier(Supplier\_ID [PK], Name, Contact\_Person, Phone, Email)

**8. Relational Algebra (RA) Queries.**

**1. List all customers**

**π CustomerID, Name, Email, Phone, Address (Customer)**

**2. Find all books available in stock**

**σ(StockQuantity > 0)(Book)**

**3. Retrieve the names of all categories**

**π CategoryName (Category)**

**4. Get details of customers from Delhi**

**σ(Address = 'Delhi')(Customer)**

**5. Find books cheaper than 500**

**σ(Price < 500)(Book)**

**6. List all publishers and their contacts**

**π PublisherName, Contact (Publisher)**

**7. Retrieve all orders placed by customer with ID = 1**

**σ(CustomerID = 1)(Order)**

**8. Get payment details for OrderID = 2**

**σ(OrderID = 2)(Payment)**

**9. Find all books by "Author1"**

**σ(Author = 'Author1')(Book)**

**10. Show customer names who placed at least one order**

**π Name (Customer ⨝ Customer.CustomerID = Order.CustomerID Order)**

**11. Get books with their category names**

**π Title, Author, CategoryName**

**(Book ⨝ Book.CategoryID = Category.CategoryID Category)**

**12. Get books with their publisher names**

**π Title, Author, PublisherName**

**(Book ⨝ Book.PublisherID = Publisher.PublisherID Publisher)**

**13. Retrieve all order items with book titles**

**π OrderItemID, Title, Quantity, Subtotal**

**(OrderItem ⨝ OrderItem.BookID = Book.BookID Book)**

**14. List all orders with customer names**

**π OrderID, Name, OrderDate, Status**

**(Order ⨝ Order.CustomerID = Customer.CustomerID Customer)**

**15. Get all payments with customer names**

**π PaymentID, Name, Amount, PaymentMethod**

**(Payment ⨝ Payment.OrderID = Order.OrderID Order ⨝ Order.CustomerID = Customer.CustomerID Customer)**

**16. Find all pending orders**

**σ(Status = 'Pending')(Order)**

**17. Get all customers who have not placed an order (anti-join)**

**π CustomerID, Name (Customer) − π CustomerID, Name (Customer ⨝ Order)**

**18. Find orders along with their total amount**

**π OrderID, TotalAmount (Order)**

**19. Find order items for a given OrderID = 1**

**σ(OrderID = 1)(OrderItem)**

**20. Find books with price between 300 and 700**

**σ(Price ≥ 300 ∧ Price ≤ 700)(Book)**

**21. Get customers and their order IDs (even if no order – left outer join)**

**Customer ⟕ Customer.CustomerID = Order.CustomerID Order**

**22. Count total orders for each customer**

**γ CustomerID; COUNT(OrderID)→TotalOrders (Order)**

**23. Find top-selling books by quantity**

**γ BookID; SUM(Quantity)→TotalSold (OrderItem)**

**24. Find total revenue of each order**

**γ OrderID; SUM(Subtotal)→TotalAmount (OrderItem)**

**25. Find total revenue generated by each publisher**

**γ PublisherID; SUM(Subtotal)→TotalRevenue**

**(OrderItem ⨝ OrderItem.BookID = Book.BookID Book)**