**Project Title:** Online Bookstore Data Management System using PostgreSQL

**Duration:** 1 Month

**Location:** Remote

**Positition:** SQL Developer Intern

**Submitted by: Hardik Rathod**

  
**Platform:** Elevate Labs

## **1. **Introduction****

In the digital era, online bookstores have become a convenient platform for readers to purchase books globally. Managing a large inventory, tracking customer orders, and maintaining sales data manually is inefficient. This project introduces a backend database system for an **Online Bookstore**, developed using **PostgreSQL**, which is known for its robustness and SQL compliance.

The system handles three major entities:

**Books** (inventory, genre, price, etc.)

**Customers** (location, identity)

**Orders** (transactional data)

The main aim is to structure the data relationally and derive meaningful insights using SQL queries. From tracking best-selling books to monitoring customer behavior, this project forms the foundation of any e-commerce book platform.

## 2. ****Objective****

The core objectives of this project are as follows:

**Database Design**:

To design a normalized relational schema using **PostgreSQL** that supports referential integrity and avoids redundancy.

**Data Management via SQL**:

To perform Create, Read, Update, Delete (CRUD) operations and write meaningful **SQL queries** for real-world use cases.

**Data Analysis and Business Insights**:

To analyze total sales, inventory stock, customer purchasing patterns, and genre-specific trends using aggregation and filtering.

**Scalable Design for Future Use**:

To structure the database in a way that it can be extended with a web interface, visual dashboards, and additional features like payments or delivery tracking.

## 3. ****Tools & Technologies****

| **Tool / Technology** | **Purpose** |
| --- | --- |
| **PostgreSQL** | Open-source RDBMS used for data storage and querying |
| **pgAdmin / DBeaver** | GUI tools used to write and test SQL queries |
| **Windows OS** | Development environment where setup was completed |

These tools collectively enabled schema creation, query development, and data testing in a structured workflow.

## 4. ****Database Design****

The system consists of three core tables:

### ****Books****

| **Field** | **Description** |
| --- | --- |
| book\_id | Primary Key |
| title | Name of the book |
| author | Name of the author |
| genre | Fiction, Fantasy, etc. |
| price | Book price |
| published\_year | Year of publication |
| stock | Available quantity |

### ****Customers****

| **Field** | **Description** |
| --- | --- |
| customer\_id | Primary Key |
| name | Full name of customer |
| city | City of residence |
| country | Country of origin |

### ****Orders****

| **Field** | **Description** |
| --- | --- |
| order\_id | Primary Key |
| customer\_id | Foreign Key referencing customers |
| book\_id | Foreign Key referencing books |
| order\_date | Date of purchase |
| quantity | Units purchased |
| total\_amount | price × quantity |

### ****Relationships****

orders.customer\_id → customers.customer\_id

orders.book\_id → books.book\_id  
This ensures referential integrity and smooth JOIN operations.

## 5. ****SQL Query Implementation****

A combination of **basic** and **advanced SQL queries** was implemented to extract useful data patterns.

### Basic Queries:

Books filtered by genre (e.g., Fiction)

Books published after a specific year (e.g., 1950)

Customers from specific locations (e.g., Canada)

Orders from a given time period (e.g., November 2023)

High-value orders (amount > $20)

Most expensive books

Total stock across all books

Total revenue from all orders

### Advanced Queries:

**Genre-wise sales**: Total quantity of books sold per genre

**Loyal Customers**: Customers who placed more than one order

**Inventory tracking**: Remaining stock after fulfilling orders

**Geographical insights**: Cities where high-spending customers are located

**Top Authors**: Based on number of books sold

**Top Fantasy Books**: 3 most expensive books in Fantasy genre

These queries were tested using **sample records** and refined using JOINs, aggregation functions (SUM, COUNT, AVG), and filtering conditions.

## 

## 6. ****Business Insights****

The data retrieved through SQL queries helped derive the following key insights:

| **Insight** | **Description** |
| --- | --- |
| 📚 **Most Sold Genre** | Helps identify the genre with highest sales for inventory planning and marketing focus |
| 🛍️ **Top Buyers** | Allows the business to recognize loyal and high-paying customers for potential rewards |
| 🏬 **High-Selling Cities** | Identifies regions with higher demand to run local ads and promotions |
| 💰 **Total Revenue** | Provides a direct view of sales performance and monthly earnings |
| 📉 **Low Stock Books** | Assists in reordering books before they go out of stock |
| 🖋️ **Author Popularity** | Authors with maximum books sold can be promoted for visibility and partnerships |

## 7. ****Conclusion****

This project successfully achieved its goal of designing and querying a **PostgreSQL-based backend system** for an online bookstore. The relational model ensured data integrity, while SQL queries allowed exploration of valuable business insights.

Key takeaways:

Clean and normalized schema design

Complex queries using joins and aggregations

Real-life use case simulation of a retail system

A foundation for building a full-fledged web-based bookstore application

## 8. ****Future Enhancements****

The current system can be extended further by integrating the following features:

**User Interface (UI):**

Frontend with **Django**, **React**, or **Flask** to allow customer interaction and admin control.

**Data Visualization:**

Dashboards using **Power BI**, **Tableau**, or **Matplotlib** to show real-time sales, top authors, and low stock alerts.

**Authentication System:**

Add login/registration system with roles (Admin/Customer) to protect sensitive operations.

**Payment Gateway Integration:**

Integrate **Razorpay**, **PayPal**, or Stripe to handle payments within the system.

**Shipping & Delivery Tracking:**

Extend order table to include delivery status, tracking number, and estimated delivery