SQL Project Report

Bookstore Database Management System

Author: Ritesh Yadav
Date: August 25, 2025

1. Introduction

This project centers on the design and implementation of a relational database for a bookstore. The primary objective is to create an efficient system to manage the bookstore's inventory, track customer information, and process sales orders. By centralizing this data, the system enables powerful querying to extract business insights, monitor sales trends, and manage stock levels effectively.

2. Database Design

The database is structured around three core entities: Books, Customers, and Orders. This normalized design minimizes data redundancy and ensures data integrity.

- **Entities & Relationships:**
  - **Books:** Stores all details related to the book inventory, such as title, author, genre, and price.
  - **Customers:** Contains information about each customer, including their name and contact details.
  - **Orders:** Acts as a transactional table linking Books and Customers. It records which customer ordered which book, the quantity, and the total amount for each transaction. A single customer can place multiple orders, and a single book can be part of many orders.
- **Database Schema:**
  - **Books** (<u>Book_ID</u> SERIAL PK, Title VARCHAR(100), Author VARCHAR(100), Genre VARCHAR(50), Published_Year INT, Price NUMERIC(10, 2), Stock INT)
  - **Customers** (<u>Customer_ID</u> SERIAL PK, Name VARCHAR(100), Email VARCHAR(100), Phone VARCHAR(15), City VARCHAR(50), Country VARCHAR(150))
  - **Orders** (<u>Order_ID</u> SERIAL PK, *Customer_ID* INT FK, *Book_ID* INT FK, Order_Date DATE, Quantity INT, Total_Amount NUMERIC(10, 2))

3. Implementation

The database was implemented using SQL. The following Data Definition Language (DDL) scripts were used to create the table structures and define the relationships between them.

```
-- Create the Books Table
CREATE TABLE Books (
    Book_ID SERIAL PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(100),
    Genre VARCHAR(50),
    Published_Year INT,
    Price NUMERIC(10, 2),
    Stock INT
);

-- Create the Customers Table
CREATE TABLE Customers (
    Customer_ID SERIAL PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(15),
    City VARCHAR(50),
    Country VARCHAR(150)
);

-- Create the Orders Table
CREATE TABLE Orders (
    Order_ID SERIAL PRIMARY KEY,
    Customer_ID INT REFERENCES Customers(Customer_ID),
    Book_ID INT REFERENCES Books(Book_ID),
    Order_Date DATE,
    Quantity INT,
    Total_Amount NUMERIC(10, 2)
);
```

4. Data Analysis & Queries

Several SQL queries were executed to extract meaningful insights from the database. Below are selected examples that demonstrate the system's capabilities.

- **Query 1: Find the most frequently ordered book.**
  - This query helps identify the bookstore's best-selling item by counting its appearances in the orders.

```
SELECT b.title, COUNT(o.order_id) AS ORDER_COUNT
FROM orders o
JOIN books b ON o.book_id = b.book_id
GROUP BY b.title
ORDER BY ORDER_COUNT DESC
LIMIT 1;
```

- **Query 2: List customers who have placed at least 2 orders.**
  - This is useful for identifying loyal customers who could be targeted for marketing campaigns.

```
SELECT c.name, COUNT(o.Order_id) AS ORDER_COUNT
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.name
HAVING COUNT(Order_id) >= 2;
```

- **Query 3: Find the customer who has spent the most.**
  - This query identifies the most valuable customer based on their total spending.

```
SELECT c.name, SUM(o.total_amount) AS Total_Spent
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.name
ORDER BY Total_spent DESC
LIMIT 1;
```

- **Query 4: Calculate the total number of books sold for each genre.**
  - This provides insight into which genres are most popular among customers, helping with inventory and purchasing decisions.

```
SELECT b.Genre, SUM(o.Quantity) AS Total_Books_sold
FROM Orders o
JOIN Books b ON o.book_id = b.book_id
GROUP BY b.Genre;
```

- **Query 5: Calculate the remaining stock after all orders.**
    - This is a crucial query for inventory management, showing the current stock level for each book after accounting for all sales.

```
SELECT
   b.title,
   b.stock,
   COALESCE(SUM(o.quantity), 0) AS Order_quantity,
   b.stock - COALESCE(SUM(o.quantity), 0) AS Remaining_Quantity
FROM books b
LEFT JOIN orders o ON b.book_id = o.book_id
GROUP BY b.book_id
ORDER BY b.book_id;
```

5. Conclusion

This project successfully demonstrates the creation and application of a relational database for a bookstore. The system effectively stores and manages data for books, customers, and orders. The executed queries prove the database's ability to answer critical business questions related to sales performance, customer behavior, and inventory status.

Future enhancements could include implementing stored procedures for common operations, creating user roles for security, or developing a front-end application for user-friendly data interaction.