# Project Title

**Online Bookstore Database Analysis using SQL**

---

## ➢ Project Objective

To design and analyze an Online Bookstore database using SQL by creating relational tables and writing analytical queries to extract business insights such as sales performance, customer behavior, inventory status, and revenue trends.

---

## ➢ Database Design

The database consists of **three related tables**:

- **Books** – Stores book details and inventory
- **Customers** – Stores customer information
- **Orders** – Stores transaction and purchase data

Relationships:

- One customer → many orders
- One book → many orders

## ➢ Database Schema

```
CREATE DATABASE OnlineBookstore;
\c OnlineBookstore;

DROP TABLE IF EXISTS Orders;
DROP TABLE IF EXISTS Customers;
DROP TABLE IF EXISTS Books;

CREATE TABLE Books (
    Book_ID SERIAL PRIMARY KEY,
    Title VARCHAR(100) NOT NULL,
    Author VARCHAR(100),
    Genre VARCHAR(50),
    Published_Year INT,
```

```
    Price NUMERIC(10,2) CHECK (Price > 0),
    Stock INT CHECK (Stock >= 0)
);

CREATE TABLE Customers (
    Customer_ID SERIAL PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Phone VARCHAR(15),
    City VARCHAR(50),
    Country VARCHAR(150)
);

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 CHECK (Quantity > 0),
    Total_Amount NUMERIC(10,2) CHECK (Total_Amount > 0)
);
```

# 📌 Business Queries & Analysis

## 1. Books in Fiction genre

```
SELECT * FROM Books
WHERE Genre = 'Fiction';
```

## 2. Books published after 1950

```
SELECT * FROM Books
WHERE Published_Year > 1950;
```

## 3. Customers from Canada

```
SELECT * FROM Customers
WHERE Country = 'Canada';
```

## 4. Orders placed in November 2023

```
SELECT * FROM Orders
WHERE Order_Date BETWEEN '2023-11-01' AND '2023-11-30';
```

## 5. Total stock available

```
SELECT SUM(Stock) AS Total_Stock
FROM Books;
```

## 6. Most expensive book

```
SELECT * FROM Books
ORDER BY Price DESC
LIMIT 1;
```

## 7. Customers who ordered more than 1 quantity

```
SELECT DISTINCT c.Customer_ID, c.Name
FROM Orders o
JOIN Customers c ON o.Customer_ID = c.Customer_ID
WHERE o.Quantity > 1;
```

## 8. Orders with total amount > $20

```
SELECT * FROM Orders
WHERE Total_Amount > 20;
```

## 9. Available book genres

```
SELECT DISTINCT Genre FROM Books;
```

## 10. Book with lowest stock

```
SELECT * FROM Books
ORDER BY Stock ASC
LIMIT 1;
```

## 11. Total revenue generated

```
SELECT SUM(Total_Amount) AS Total_Revenue
FROM Orders;
```

## 12. Total books sold per genre

```
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;
```

## 13. Average price of Fantasy books

```
SELECT AVG(Price) AS Average_Price
FROM Books
WHERE Genre = 'Fantasy';
```

## 14. Customers with at least 2 orders

```
SELECT c.Customer_ID, 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.Customer_ID, c.Name
HAVING COUNT(o.Order_ID) >= 2;
```

## 15. Most frequently ordered book

```
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;
```

## 16. Top 3 most expensive Fantasy books

```
SELECT * FROM Books
WHERE Genre = 'Fantasy'
ORDER BY Price DESC
LIMIT 3;
```

## 17. Total books sold by each author

```
SELECT b.Author, SUM(o.Quantity) AS Total_Books_Sold
FROM Orders o
JOIN Books b ON o.Book_ID = b.Book_ID
GROUP BY b.Author;
```

## 18. Cities where customers spent over $30

```
SELECT DISTINCT c.City
FROM Orders o
JOIN Customers c ON o.Customer_ID = c.Customer_ID
WHERE o.Total_Amount > 30;
```

## 19. Customer with highest spending

```
SELECT c.Customer_ID, 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.Customer_ID, c.Name
ORDER BY Total_Spent DESC
LIMIT 1;
```

## 20. Remaining stock after fulfilling orders

```
SELECT b.Book_ID, b.Title, b.Stock,
    COALESCE(SUM(o.Quantity),0) AS Sold_Quantity,
    b.Stock - COALESCE(SUM(o.Quantity),0) AS Remaining_Stock
FROM Books b
LEFT JOIN Orders o ON b.Book_ID = o.Book_ID
GROUP BY b.Book_ID, b.Title, b.Stock
ORDER BY b.Book_ID;
```