PROJECT STRUCTURE

1. Project Overview

This project involves designing and implementing a relational database for an online bookstore. It manages books, customers, and orders. The system supports CRUD operations, data import, and SQL queries to retrieve insights for analytics and reporting.

2. Database Design

Database Name:

- onlinebooksstore

Tables:

- Books
- Customers
- Orders

```
-- creating database --
CREATE DATABASE onlinebooksstore;
```

3. Table Structure

Books Table:

- Book_ID: SERIAL PRIMARY KEY
- Title: VARCHAR(100)
- Author: VARCHAR(100)
- Genre: VARCHAR(100)
- Published_Year: INT
- Price: NUMERIC(10,2)
- Stock: INT

```
-- Create Table --
DROP TABLE IF EXISTS Books;

CREATE TABLE Books(
Book_ID SERIAL PRIMARY KEY,
Title VARCHAR(100),
Author VARCHAR(100),
Genre VARCHAR(100),
Published_Year INT,
price NUMERIC(10,2),
Stock INT
);
```

Customers Table:

- Customers_ID: SERIAL PRIMARY KEY
- Name: VARCHAR(100)
- Email: VARCHAR(100)
- Phone: VARCHAR(15)
- City: VARCHAR(50)
- Country: VARCHAR(150)

```
DROP TABLE IF EXISTS Customers;

CREATE TABLE Customers(
Customers_ID SERIAL PRIMARY KEY,
Name VARCHAR(100),
Email VARCHAR(100),
Phone VARCHAR(15),
City VARCHAR(50),
Country VARCHAR(150)
);
```

Orders Table:

- Order_ID: SERIAL PRIMARY KEY
- Customers_ID: INT (FK to Customers)
- Book_ID: INT (FK to Books)
- Order_Date: DATE
- Quantity: INT
- Total_Amount: NUMERIC(10,2)

```
DROP TABLE IF EXISTS Orders;

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

```
SELECT * FROM Books;
SELECT * FROM Customers;
SELECT * FROM Orders;
```

4. Data Import

CSV files are used to populate the database:

- Books.csv
- Customers.csv
- Orders.csv

```
-- import data into books table --
COPY Books(Book_ID,Title, Author,Genre,Published_Year,Price,Stock)
FROM 'D:\SQL Files\Books.csv'
delimiter ','
CSV HEADER;

-- imort data into customer table --
COPY Customers(Customers_ID,Name,Email,Phone,City,Country)
FROM 'D:\SQL Files\Customers.csv'
delimiter ','
CSV HEADER;

-- import data into order table --
COPY Orders(Order_ID,Customers_ID,Book_ID,Order_Date,Quantity,Total_Amount)
FROM 'D:\SQL Files\Orders.csv'
delimiter ','
CSV HEADER;
```

5. Basic Queries

1. Retrieve all books in the "Fiction" genre

```
-- 1) retrieve all books in the "Fiction " genre: --

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

2. Find books published after 1950

```
-- 2) find books published after the year 1950: --

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

3. List customers from Canada

```
-- 3) list all customers from the canada: --

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

4. Show orders placed in November 2023

```
--4) show orders placed in November 2023: --

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

5. Retrieve total stock of books available

```
-- 5) retrive the total stock of books available: --

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

6. Find details of the most expensive book

```
-- 6) find the details of most expensive book: --

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

7. Show customers who ordered more than one quantity

```
-- 7) show all cuatomers who ordered more than 1 quantity of a book:--

SELECT * FROM Orders
WHERE Quantity>1;
```

8. Retrieve all orders where total amount exceeds $20

```
--8) retrieve all orders where the toatl amount exceeds $20"--

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

9. List all available genres

```
-- 9) list all genres available in the books table: --

SELECT DISTINCT genre FROM Books;
```

10. Find the book with the lowest stock

```sql
-- 10) find all the book with lowest stock: --

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

11. Calculate total revenue from all orders

```sql
-- 11) calculate the total revenue generated from all orders: --

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

6. Advanced Queries

1. Total number of books sold for each genre

```sql
-- 1) retrieve the total number of books sold for each 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;
```

2. Average price of books in the "Fantasy" genre

```sql
-- 2) find the average price of books in the "fantasy" genre: --

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

3. Customers who placed at least two orders

```sql
-- 3) list customers who have placed at least 2 orders: --

SELECT o.Customers_ID, c.Name, COUNT(o.Order_ID) AS ORDER_COUNT
FROM Orders o
JOIN Customers c ON o.Customers_ID = c.Customers_ID
GROUP BY o.Customers_ID, c.NAME
HAVING COUNT(Order_ID)>=2;
```

4. Most frequently ordered book

```sql
-- 4) find the most frequesntly ordered book: --

SELECT o.Book_ID, b.Title, COUNT(o.Order_ID) AS ORDER_COUNT
FROM Orders o
JOIN Books b ON o.Book_ID = b.Book_ID
GROUP BY o.Book_ID,b.Title
ORDER BY ORDER_COUNT DESC
LIMIT 1;
```

5. Top 3 most expensive books in the "Fantasy" genre

```sql
-- 5) show the top 3 most expensive books of 'Fantasy' Genre: --

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

6. Total quantity of books sold by each author

```sql
-- 6) retrieve the total quantity of 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;
```

7. Cities where customers who spent over $30 are located

```sql
-- 7) list of cities where customers who spent over $30 are located:--

SELECT c.city, Total_Amount
FROM Orders o
JOIN Customers c ON o.Customers_ID = c.Customers_ID
WHERE o.Total_Amount>30;
```

8. Customer who spent the most on orders

```sql
-- 8) find the customer who spent most on orders: --

SELECT c.Customers_ID, c.Name, SUM(Total_Amount) AS Total_Spent
FROM Orders o
JOIN Customers c ON o.Customers_ID = c.Customers_ID
GROUP BY c.Customers_ID, c.Name
ORDER BY Total_Spent DESC
LIMIT 1;
```

9. Remaining stock after fulfilling orders

```sql
-- 9) calculate the stock remaining after fulfilling all orders: --

SELECT b.Book_ID, 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;
```

7.ER Diagram: