SQL Project

Online Book Store

Data Tables: Books.csv, Customers.csv, Orders.csv

```
--create Database--
CREATE DATABASE onlinebookstore


--create tables--
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 TABLE Customers (
    Customer_ID SERIAL PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    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,
    Total_Amount NUMERIC(10, 2)
);

SELECT * FROM Books
SELECT * FROM Customers
SELECT * FROM Orders

--import table Books--
COPY Books(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)
FROM 'C:\Program Files\PostgreSQL\Online_bookstore\Books.csv'
CSV HEADER;

--import table Customers--
COPY Customers(Customer_ID, Name, Email, Phone, City, Country)
FROM 'C:\Program Files\PostgreSQL\Online_bookstore\Customers.csv'
CSV HEADER;

--import table Orders--
COPY Orders(Order_ID, Customer_ID, Book_ID, Order_Date, Quantity,
Total_Amount)
```

```
FROM 'C:\Program Files\PostgreSQL\Online_bookstore\Orders.csv'
CSV HEADER;

--QUERIES--

-- 1) Retrieve all books in the "Fiction" genre:
SELECT * FROM Books
WHERE Genre = 'Fiction';

-- 2) Find books published after the year 1950:
SELECT * FROM Books
WHERE Published_Year > '1950';

-- 3) List all customers from the Canada:
SELECT * FROM Customers
WHERE Country = 'Canada';

-- 4) Show orders placed in November 2023:
SELECT * FROM Orders
WHERE Order_Date BETWEEN '2023-11-01' AND '2023-11-30';

-- 5) Retrieve the total stock of books available:
SELECT SUM(Stock) AS Total_Stock FROM Books;

-- 6) Find the details of the most expensive book:
SELECT * FROM Books
```

```
ORDER BY Price DESC
LIMIT 1;

-- 7) Show all customers who ordered more than 1 quantity of a book:
SELECT * FROM Orders
WHERE Quantity > 1;

-- 8) Retrieve all orders where the total amount exceeds $20:
SELECT * FROM Orders
WHERE Total_Amount > 20;

-- 9) List all genres available in the Books table:
SELECT Genre FROM Books;

-- 9) List all genres available in the Books table:
SELECT DISTINCT Genre FROM Books;

-- 10) Find the book with the lowest stock:
SELECT * FROM Books
ORDER BY Stock
LIMIT 1;

-- 11) Calculate the total revenue generated from all orders:
SELECT SUM(Total_Amount) AS Revenue FROM Orders;

-- Advance Questions :
```

```
-- 1) Retrieve the total number of books sold for each genre:
SELECT b.Genre, SUM(o.Quantity) AS Total_book_sold
FROM Orders o
JOIN Books b ON b.book_id = o.book_id
GROUP BY b.Genre;

-- 2) Find the average price of books in the "Fantasy" genre:
SELECT AVG(Price) AS Average_Price
FROM Books
WHERE Genre = 'Fantasy';

-- 3) List customers who have placed at least 2 orders:
SELECT o.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 o.Customer_id, c.name
HAVING COUNT(Order_id) >= 2;

-- 4) Find the most frequently ordered book:
SELECT b.Book_id, b.Title, COUNT(o.Order_id) AS Order_Count
FROM Orders o
JOIN Books b ON b.book_id = o.book_id
GROUP BY b.Book_id
ORDER BY Order_Count DESC
LIMIT 1;
```

```
-- 5) Show the top 3 most expensive books of 'Fantasy' Genre :
SELECT * FROM Books
WHERE GENRE = 'Fantasy'
ORDER BY Price DESC
LIMIT 3;

-- 6) Retrieve the total quantity of books sold by each author:
SELECT b.Author, SUM(o.quantity) AS Total_Quantity
FROM Orders o
JOIN Books b ON b.book_id = o.book_id
GROUP BY b.Author;

-- 7) List the cities where customers who spent over $30 are located:
SELECT DISTINCT(c.city), O.Total_Amount
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
WHERE o.total_amount > 30;

-- 8) Find the customer who spent the most on orders:
SELECT  c.customer_id, c.name, SUM(o.total_amount) AS Total_Spent
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name
ORDER BY total_spent DESC
LIMIT 1;
```

```
--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_Order

FROM Books b

LEFT JOIN Orders o ON b.book_id = o.book_id

GROUP BY b.book_id

ORDER BY b.book_id ASC;
```