```sql
-- Create Database Online_Bookstore
-- Create Tables
DROP TABLE IF EXISTS Books;
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
);
DROP TABLE IF EXISTS customers;
CREATE TABLE Customers (
    Customer_ID SERIAL PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(15),
    City VARCHAR(50),
    Country VARCHAR(150)
);

DROP TABLE IF EXISTS orders;
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 Data into Books Table
COPY Books(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)
FROM 'D:\Data Analysis Projects\Online Bookstore Analysis Uisng Postgre
SQL\Books.csv'
CSV HEADER;



-- Import Data into Customers Table
COPY Customers(Customer_ID, Name, Email, Phone, City, Country)
FROM 'D:\Data Analysis Projects\Online Bookstore Analysis Uisng Postgre
SQL\Customers.csv'
CSV HEADER;

-- Import Data into Orders Table
COPY Orders(Order_ID, Customer_ID, Book_ID, Order_Date, Quantity,
Total_Amount)
FROM 'D:\Data Analysis Projects\Online Bookstore Analysis Uisng Postgre
SQL\Orders.csv'
CSV HEADER;
```

```sql
-- 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 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 * from orders

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) Find the average price of books in the "Fantasy" genre:
select avg(price) as Avg_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 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) 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_books_sold
from orders o
join books b on o.book_id = b.book_id
group by b.author;

-- 7) List the cities where customers who spent over $30 are located:
select distinct c.city, 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 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;

--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;
```