Online Book Store

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

--Create 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
);


--Drop Table
DROP TABLE IF EXISTS Customers;

--Create Table
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

```
DROP TABLE IF EXISTS Orders;
```

--Create 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)
);
```

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

--Basics Query

--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;
```

--Advanced Query

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

--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 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 customer_id, COUNT(ORDER_ID) AS order_count
FROM Orders
GROUP BY CUSTOMER_ID
HAVING COUNT(ORDER_ID)>=2;
```

OR

```
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 BOOK_ID, COUNT(ORDER_ID) AS order_count
FROM Orders
GROUP BY BOOK_ID
ORDER BY order_count DESC
LIMIT 1;
```

OR

```
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_book_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;
```

--9) Calculate the stock remaining after fulfilling all orders

```
SELECT b.book_id, b.title, b.stock, COALESCE(SUM(quantity),0) AS order_quantity,
        b.stock - COALESCE(SUM(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;
```