ONLINE_BOOK_STORE DATABASE

```
DROP TABLE IF EXISTS books;
CREATE TABLE Books(
                Book_ID SERIAL PRIMARY KEY,
                Title  VARCHAR(100),
                Author VARCHAR(50),
                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)
);

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;

-- BASIC QUERIES:
-- 1. Retrieve all books in the "FICTION" genre
            SELECT * FROM Books
            WHERE genre = 'Fiction';
```

-- 2. Find books published after the year 1950

```
SELECT title , published_year 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 '2023-09-01' AND '2023-09-30';
```

-- 5. Retrieve the total stock of books available

```
SELECT SUM(stock) AS TOTAL_BOOKS_AVILABLE
 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 then 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 book table

```
SELECT DISTINCT genre FROM BOOKS
```

-- 10. Find the book with lowest stock

```
SELECT * FROM Books ORDER BY stock LIMIT 2
```

-- 11. Calculate the total revenue generated from all orders.

```
SELECT SUM(total_amount) AS revenue FROM Orders
```

-- Advance Queries:

-- 1. Retrieve the total numbers 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. Find the average price of books in the "Fantasy" genre.

```
SELECT  AVG(price) AS AVERAGE_PRICE
FROM Books
WHERE genre = 'Fantasy'
```

-- 3. List all 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;
```

— Another query for : List all customers who have placed at least 2 orders.

```
-- SELECT c.name, o.quantity AS ALL_CUSTOMERS_2ORD
-- FROM Orders o
-- JOIN Customers c ON o.customer_id = c.customer_id
-- WHERE quantity >1 ORDER BY quantity
```

-- 4. Find the SECOND 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 OFFSET 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.**

```
SELECT 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 customers 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(o.quantity),0) AS
order_quantity,
b.stock - COALESCE(SUM(o.quantity),0) AS TOTAL_REMEANING
FROM books b
LEFT JOIN orders o ON b.book_id = o.book_id
GROUP BY b.book_id ORDER BY b.book_id;
```