```
show databases;

-- create table if exists Books;

-- create table if not 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

);


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)

);
```

```
-- IMPORT DATA

SELECT * FROM Books;

SELECT * FROM Customers;

SELECT * FROM Orders;



ALTER TABLE BOOKS

RENAME COLUMN published_year to publishe_year;



-- truncate table books;

-- truncate table employee restart identity;


UPDATE Books

SET

   price = 65.52

WHERE

   book_id = 3


-- Import Data into Books Table

-- Import Data into Customers Table

-- Import Data into Orders Table



-- 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;



SELECT
   *
FROM
   Books
WHERE
   Published_year > 1950;




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



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

```sql
SELECT
  *
FROM
  Orders
WHERE
  order_date BETWEEN '2023-11-1' 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
   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 customers who have placed at least 2 orders:
SELECT
   *
FROM
   orders;
SELECT
   customer_id, COUNT(order_id) AS order_count
FROM
   orders
GROUP BY customer_id
HAVING COUNT(order_id) >= 2;
```

```
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;



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 c.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;
```