```sql
CREATE DATABASE sql_proj_01;
USE sql_proj_01;

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 * 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 WHERE price = (SELECT MAX(price) FROM books);

SELECT TOP 1 * FROM books ORDER BY price DESC;

-- 7) Show all customers who ordered more than 1 quantity of a book
SELECT * FROM customers WHERE customer_id IN
(SELECT customer_id 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 WHERE stock = (SELECT MIN(stock) FROM books);

SELECT TOP 1 * FROM books ORDER BY stock ASC; -- only one

-- 11) Calculate the total revenue generated from all orders
SELECT SUM(total_amount) AS total_revenue FROM orders;




-- Advance Queries
-- 1) Retrieve the total number of books sold for each genre
WITH cte AS
(SELECT book_id, SUM(quantity) AS books_sold FROM orders GROUP BY book_id)
```

```sql
SELECT b.genre, SUM(cte.books_sold) AS total_books_sold
FROM books b INNER JOIN cte ON b.book_id = cte.book_id
GROUP BY b.genre;

-- 2) Find the average price of books in the "Fantasy" genre
SELECT genre, AVG(price) AS avg_price FROM books GROUP BY genre HAVING genre =
  'Fantasy';

SELECT genre, AVG(price) AS avg_price FROM books WHERE genre = 'Fantasy' GROUP
  BY genre;

-- 3) List customers who have placed at least 2 orders
SELECT * FROM customers WHERE customer_id IN
(SELECT customer_id FROM orders GROUP BY customer_id HAVING COUNT(customer_id) >
   1);

-- 4) Find the most frequently ordered book
WITH cte AS
(SELECT book_id, COUNT(book_id) AS ordr_cnt FROM orders GROUP BY book_id)
SELECT b.title, cte.ordr_cnt FROM cte
INNER JOIN books b ON b.book_id = cte.book_id
WHERE cte.ordr_cnt = (SELECT MAX(cte.ordr_cnt) FROM cte)


-- 5) Show the top 3 most expensive books of 'Fantasy' Genre
SELECT TOP 3 * FROM books WHERE genre = 'Fantasy' ORDER BY price DESC;

-- 6) Retrieve the total quantity of books sold by each author
WITH cte AS
(SELECT book_id, SUM(quantity) AS qty FROM orders GROUP BY book_id)
SELECT b.author, SUM(cte.qty) AS quantity FROM books b INNER JOIN cte ON
  b.book_id = cte.book_id
GROUP BY b.author;

-- 7) List the cities where customers who spent over $30 are located
WITH cte AS
(SELECT customer_id, SUM(total_amount) AS spent FROM orders GROUP BY customer_id
   HAVING SUM(total_amount) > 30)
SELECT c.city FROM customers c INNER JOIN cte ON c.customer_id =
  cte.customer_id;

-- 8) Find the customer who spent the most on orders
WITH cte AS
(SELECT customer_id, SUM(total_amount) AS spent FROM orders GROUP BY
  customer_id)
SELECT c.name FROM customers c
INNER JOIN cte ON c.customer_id = cte.customer_id
WHERE cte.spent = (SELECT MAX(cte.spent) FROM cte);

-- 9) Calculate the stock remaining after fulfilling all orders
WITH cte AS
(SELECT book_id, SUM(quantity) AS ordr_qty FROM orders GROUP BY book_id),
cte2 AS
```

```sql
(SELECT b.book_id, b.stock, cte.ordr_qty, (b.stock - ordr_qty) AS stock_rem FROM
   books b LEFT JOIN cte ON b.book_id = cte.book_id)
SELECT SUM(cte2.stock_rem) AS stock_remaining FROM cte2;
```