```
-- CREATE DATABASE

CREATE DATABASE OnlineBookStore;

USE OnlineBookStore;

-- CREATE TABLE FOR BOOKS TABLE
CREATE TABLE Books (
	Book_ID INT PRIMARY KEY,
	Title VARCHAR(100),
	 Author VARCHAR(100),
	Genre VARCHAR(50),
	Published_Year INT,
	Price NUMERIC(10,2),
	Stock INT
);

-- CREATE TABLE FOR CUSTOMER TABLE
CREATE TABLE Customers (
	Customer_ID INT PRIMARY KEY,
	 Name VARCHAR(100),
	Email VARCHAR(100),
	Phone VARCHAR(100),
	City VARCHAR(50),
	Country VARCHAR(100)
);

-- CREATE TABLE FOR ORDERS TABLE
CREATE TABLE Orders (
	Order_ID INT 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 into Table Using MySQL Workbench

-- TO VIEW ALL DATASET
```

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

-- 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 ASC
LIMIT 1;

-- 11) Calculate the total revenue generated from all orders:

SELECT SUM(Total_Amount) AS Total_Revenue
FROM Orders;

-- 12) 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;

-- 13) Find the average price of books in the "Fantasy" genre:

SELECT AVG(Price) AS Average_Price
FROM Books
WHERE Genre = 'Fantasy';

-- 14) 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 BY CUSTOMER NAME

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;

-- 15) 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;

-- 16) Show the top 3 most expensive books of 'Fantasy' Genre :

SELECT * FROM Books
WHERE Genre = "Fantasy"
ORDER BY Price DESC
LIMIT 3;

-- 17) Retrieve the total quantity of books sold by each author:

SELECT B.Author, SUM(O.Quantity) AS Books_Sold
FROM Orders O
JOIN Books B ON O.Book_ID = B.Book_ID
GROUP BY B.Author;

-- 18) List the cities where customers who spent over $30 are located

SELECT DISTINCT C.City, O.Total_Amount
FROM Orders O
JOIN Customers C ON O.Customer_ID = C.Customer_ID
WHERE O.Total_Amount > 30;
```

-- 19) 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;
```

-- 20) 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;
```

-- PROJECT FINISHED --