ONLINEBOOKSTORE DATA ANALYSIS

```
-- Already Created Database, hence creating three table i.e., BOOK, CUSTOMER and ORDERS.
CREATE TABLE BOOK(
        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 CUSTOMER(
        Customer_ID INT PRIMARY KEY,
        Name VARCHAR(100) NOT NULL,
        Email VARCHAR(150) UNIQUE,
        Phone VARCHAR(15),
        City VARCHAR(100),
        Country VARCHAR(100)
);


CREATE TABLE ORDERS(
        Order_ID INT PRIMARY KEY,
        Customer_ID INT REFERENCES CUSTOMER(Customer_ID),
        Book_ID INT REFERENCES BOOK(Book_ID),
        Order_Date DATE,
        Quantity INT,
        Total_Amount NUMERIC(10,2)
);
```

-- As there was the permission denied issue when writing the import query, imported the data for the following table manually.

```
SELECT * FROM BOOK;
SELECT * FROM CUSTOMER;
SELECT * FROM ORDERS;
```

-- Basic Queries

-- 1. Retrieve all books in the "Fiction" genre

```
SELECT * FROM BOOK
WHERE genre = 'Fiction';
```

-- 2. Find books published after the year 1950

```
SELECT * FROM BOOK
WHERE published_year>1950;
```

--3. List all customers from the Canada

```
SELECT * FROM  CUSTOMER
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 BOOK;
```

--6. Find the details of the most expensive book

```
SELECT * FROM BOOK
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 BOOK;
```

--10. Find the book with the lowest stock

```
SELECT * FROM BOOK
ORDER BY stock LIMIT 1;
```

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

```
SELECT SUM (total_amount) AS Revenue
FROM ORDERS;
```

--Advance Queries

--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 BOOK 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 BOOK
WHERE genre = 'Fantasy';
```

--3. List customers who have placed at least 2 orders

```
SELECT c.Customer_ID, c.name, COUNT(o.order_id) AS Order_Count
FROM ORDERS o
JOIN CUSTOMER c ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.name
HAVING COUNT(o.order_id) >= 2;
```

--4. Find the most frequently ordered book

```
SELECT  o.book_id,b.title, COUNT (o.order_id) AS ORDERS_COUNT
FROM ORDERS o
JOIN BOOK b ON o.book_id=b.book_id
GROUP BY o.book_id, b.title
ORDER BY ORDERS_COUNT DESC LIMIT 1;
```

--5. Show the top 3 most expensive books of 'Fantasy' Genre

```
SELECT * FROM BOOK
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 BOOK 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 CUSTOMER 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 CUSTOMER 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(quantity),0) AS Order_quantity,
                b.stock - COALESCE(SUM(o.quantity),0) AS Remaining_quantity
FROM BOOK b
LEFT JOIN ORDERS o ON b.book_id=o.book_id
GROUP BY b.book_id
ORDER BY b.book_id ASC;
```