SQL Bookstore Analysis Project Queries

Database and Table Creation

```
CREATE DATABASE bookstore;
USE bookstore;

DROP TABLE IF EXISTS Books;
CREATE TABLE Books (
    Book_ID INT NOT NULL PRIMARY KEY,
    Title VARCHAR(100) NOT NULL,
    Author VARCHAR(100) NOT NULL,
    Genre VARCHAR(50) NOT NULL, -- Assuming a reasonable length for Genre
    Published_Year INT NOT NULL,
    Price DOUBLE NOT NULL,
    Stock INT NOT NULL
);

DROP TABLE IF EXISTS Customers;
CREATE TABLE Customers (
    Customer_ID INT NOT NULL PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Phone INT NOT NULL, -- Consider VARCHAR if phone numbers contain non-
numeric characters or leading zeros
    City VARCHAR(50) NOT NULL,
    Country VARCHAR(150) NOT NULL
);

DROP TABLE IF EXISTS Orders;
CREATE TABLE Orders (
    Order_ID INT NOT NULL PRIMARY KEY,
    Customer_ID INT REFERENCES Customers (Customer_ID),
    Book_ID INT REFERENCES Books (Book_ID),
    Order_Date DATE NOT NULL,
    Quantity INT NOT NULL,
    Total_Amount DOUBLE NOT NULL
);
```

Analysis Questions and 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 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 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 100:

```
SELECT
   *
FROM
   Orders
WHERE
   total_amount > 100;
```

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
    ROUND(SUM(total_amount), 2) AS Revenue -- Assuming rounding to 2 decimal
places
FROM
    Orders;
```

12. Retrieve the total number of books sold for each genre:

```
SELECT
    b.Genre,
    SUM(o.Quantity) AS Total_Books
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 "non-fiction" genre:

```
SELECT
    ROUND(AVG(price), 2) AS Average_Price
FROM
    Books
WHERE
    Genre = 'non-fiction';
```