SQL Project on Online Book store

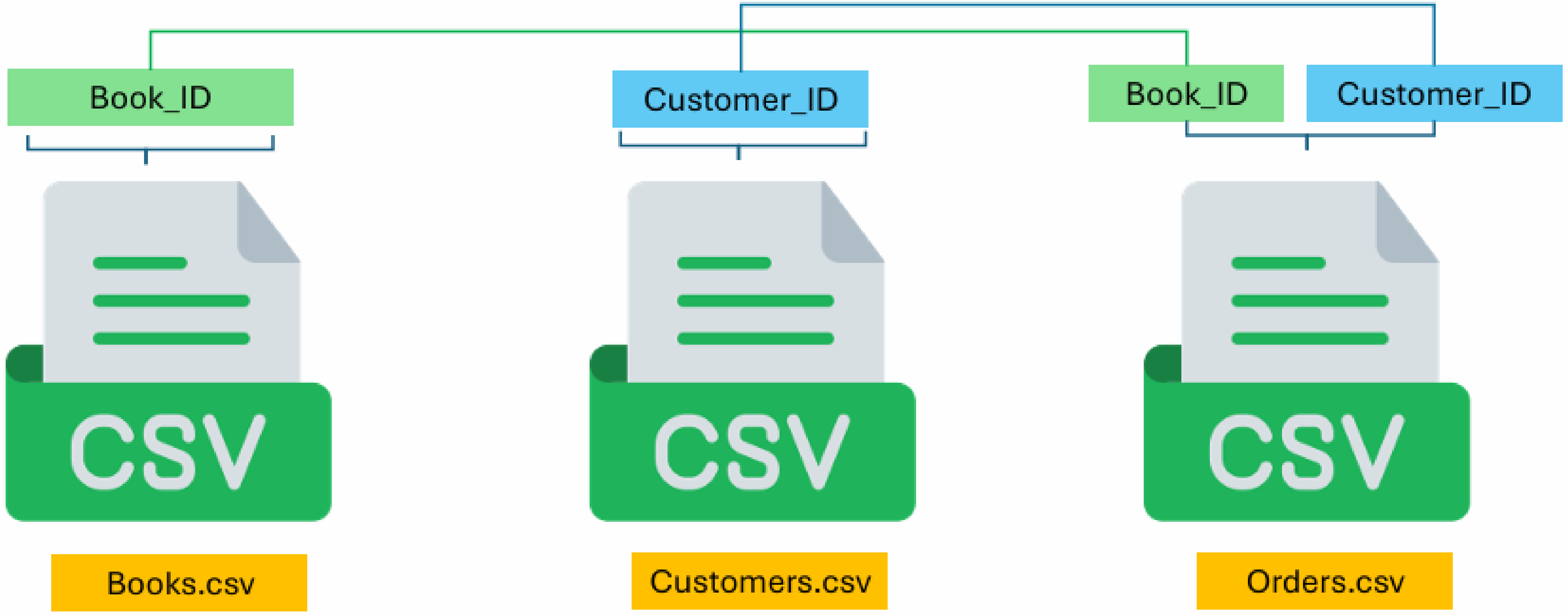
All problem statements along with their corresponding solutions have been added to the above-mentioned pages.

Presented by

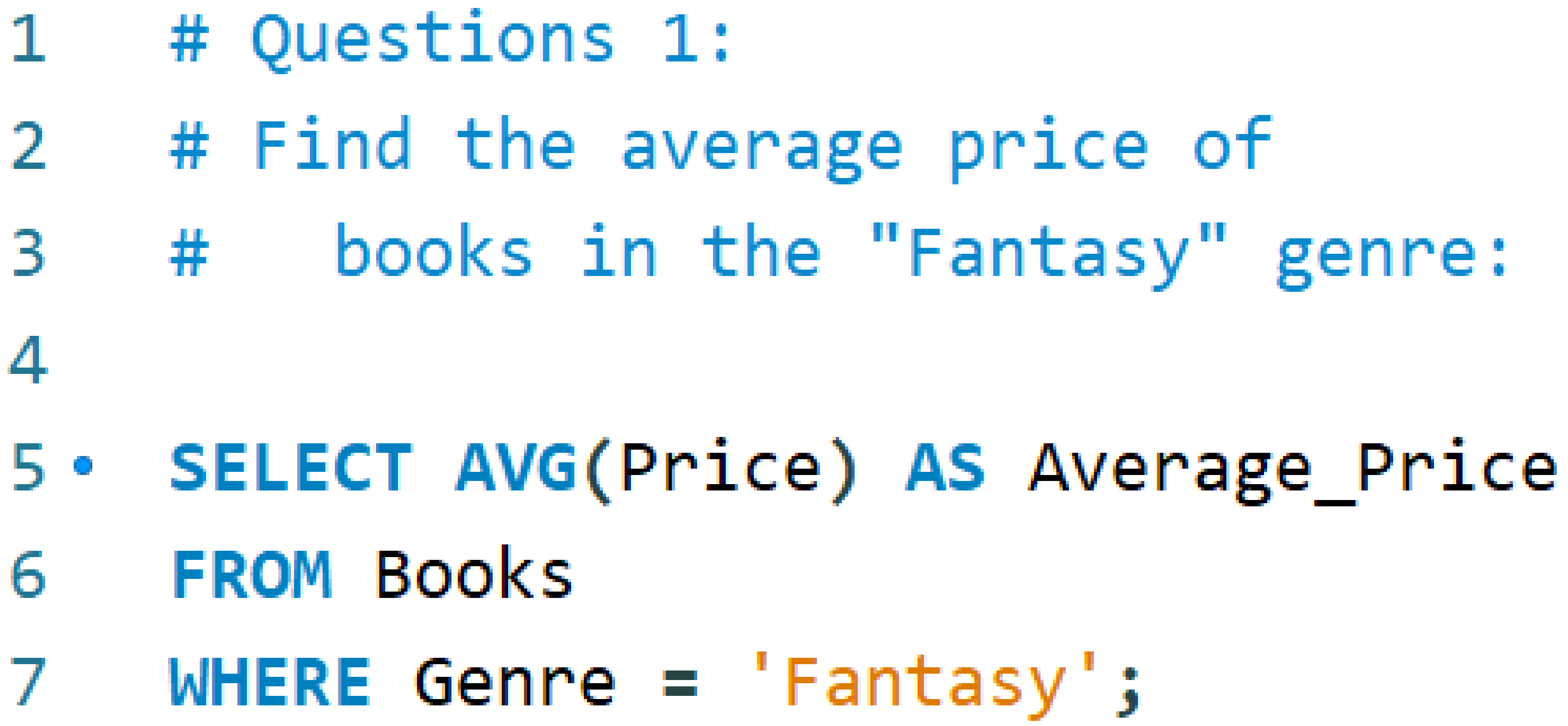
Mayank Kumar

3 CSV Files

Tables must have at least one common column with same column name and same data type



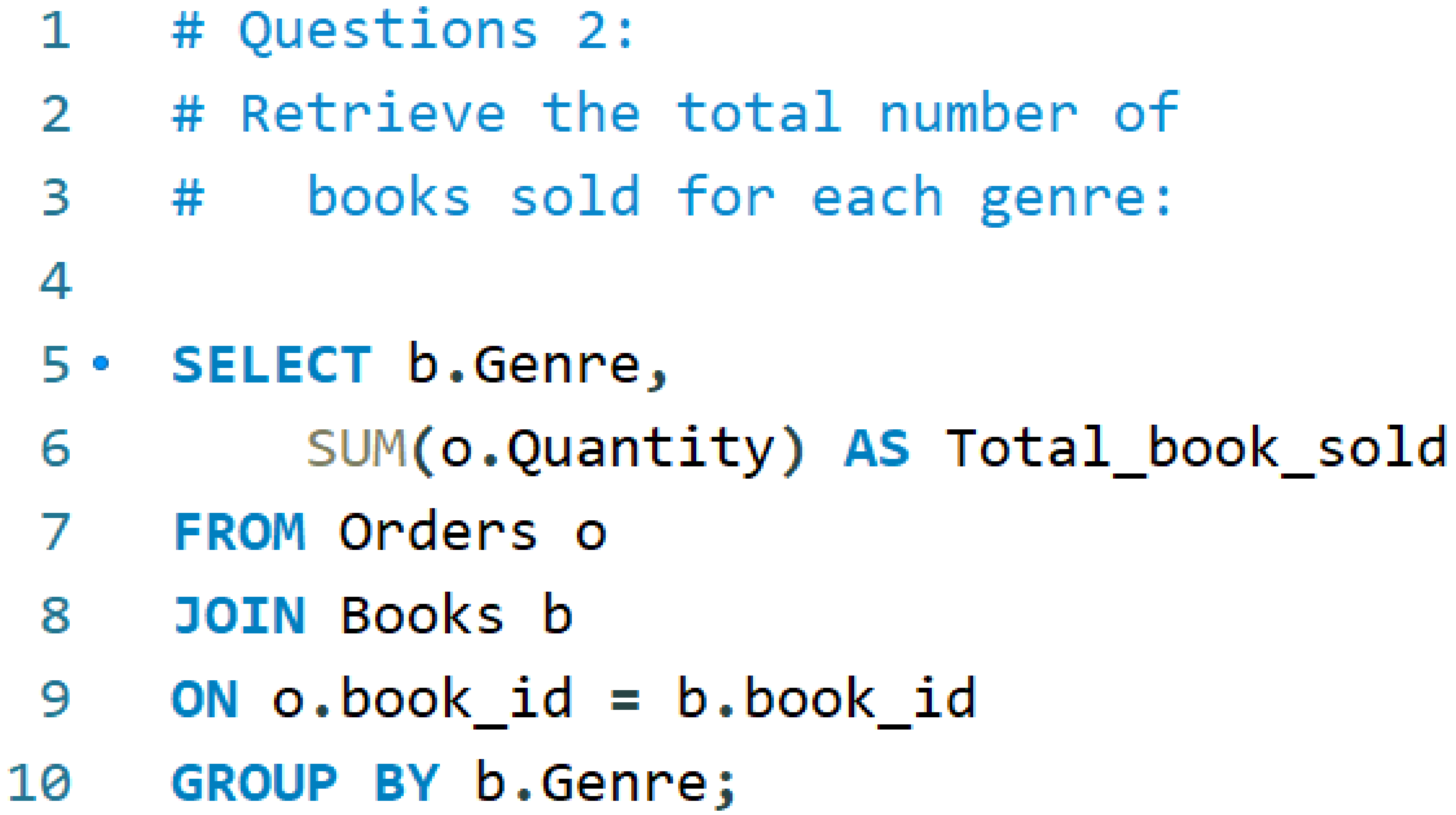
QUESTION 1



```
# Questions 1:
# Find the average price of
#   books in the "Fantasy" genre:

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

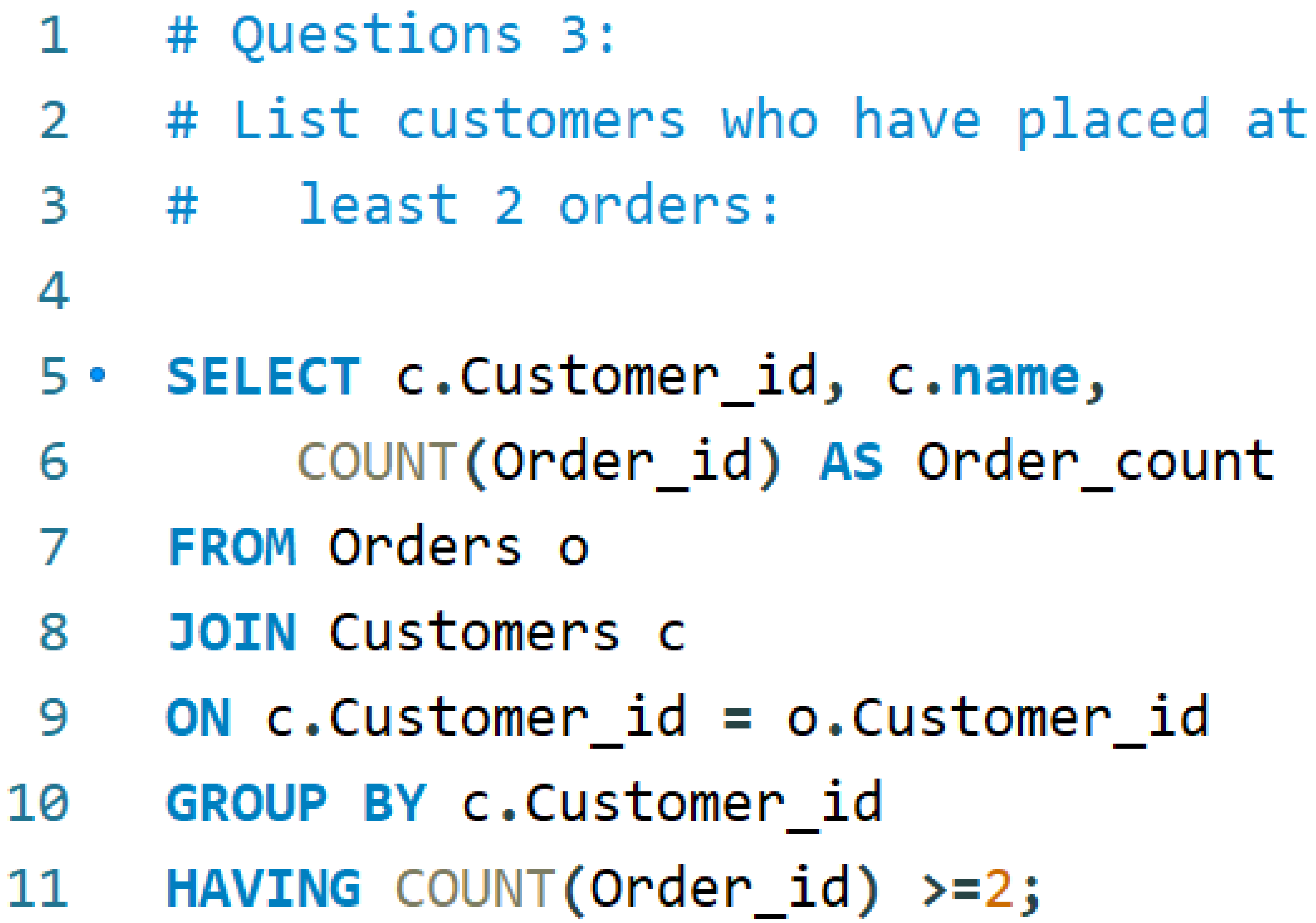
QUESTION 2



```sql
# Questions 2:
# Retrieve the total number of
#    books sold for each genre:

SELECT b.Genre,
       SUM(o.Quantity) AS Total_book_sold
FROM Orders o
JOIN Books b
ON o.book_id = b.book_id
GROUP BY b.Genre;
```

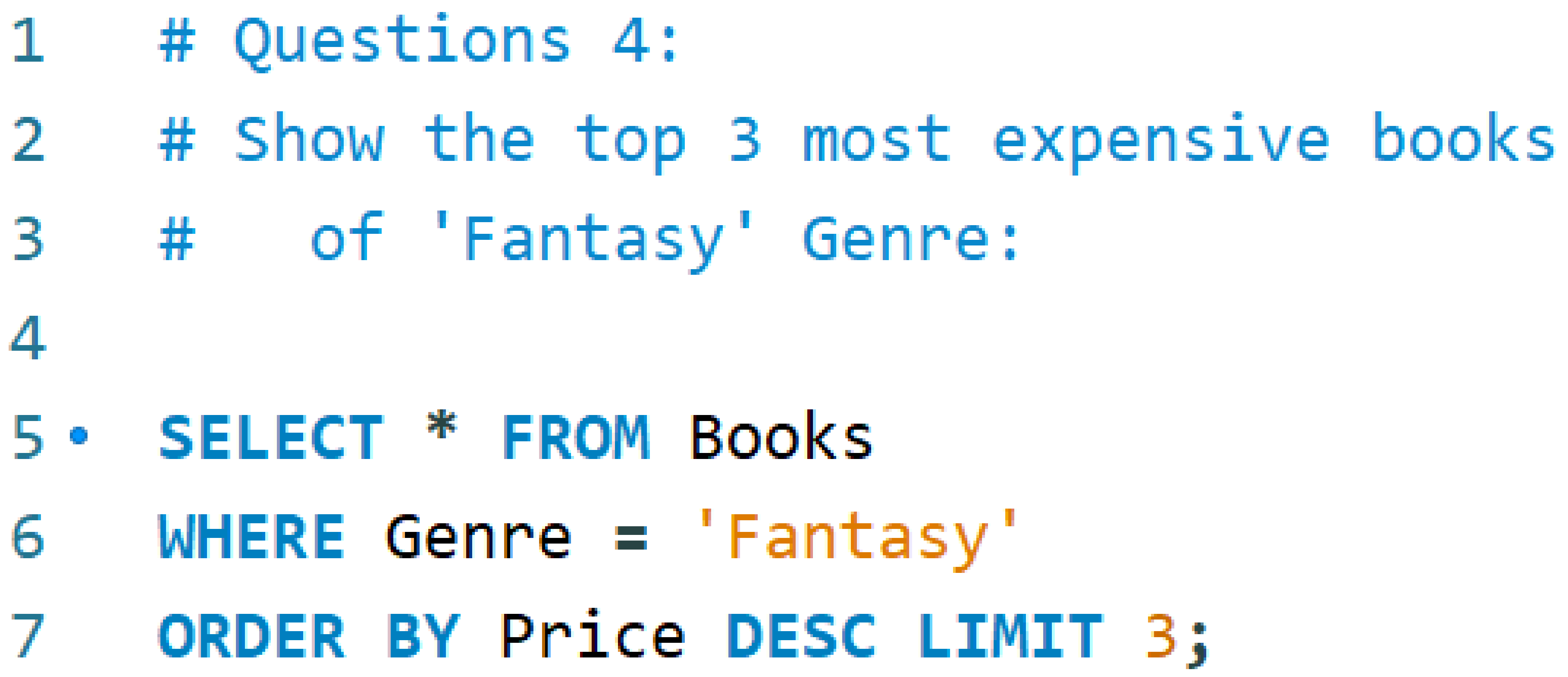
QUESTION 3



```sql
# Questions 3:
# List customers who have placed at
#    least 2 orders:

SELECT c.Customer_id, c.name,
    COUNT(Order_id) AS Order_count
FROM Orders o
JOIN Customers c
ON c.Customer_id = o.Customer_id
GROUP BY c.Customer_id
HAVING COUNT(Order_id) >=2;
```

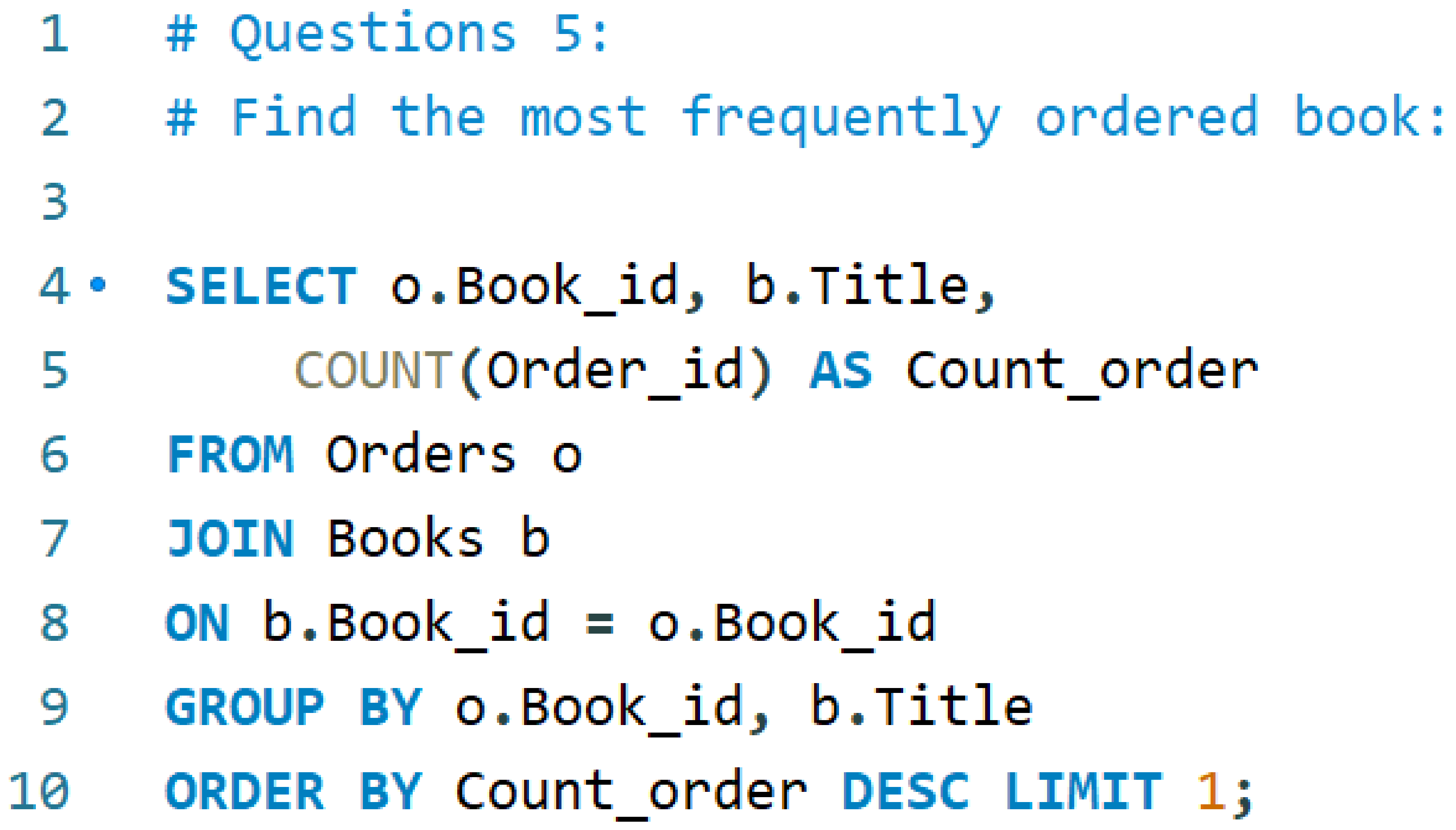
QUESTION 4



```
# Questions 4:
# Show the top 3 most expensive books
#   of 'Fantasy' Genre:

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

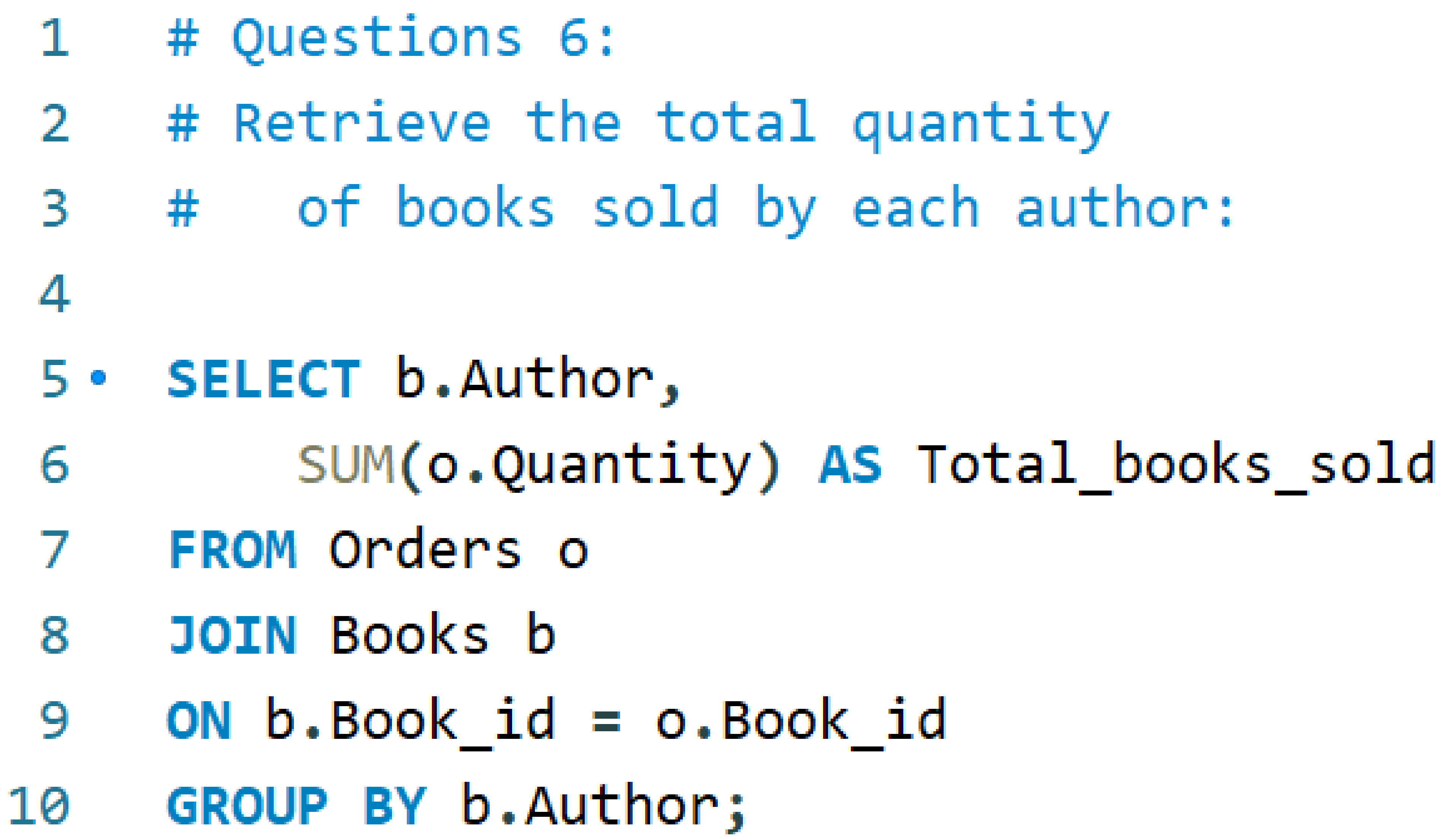
QUESTION 5



```sql
# Questions 5:
# Find the most frequently ordered book:

SELECT o.Book_id, b.Title,
       COUNT(Order_id) AS Count_order
FROM Orders o
JOIN Books b
ON b.Book_id = o.Book_id
GROUP BY o.Book_id, b.Title
ORDER BY Count_order DESC LIMIT 1;
```

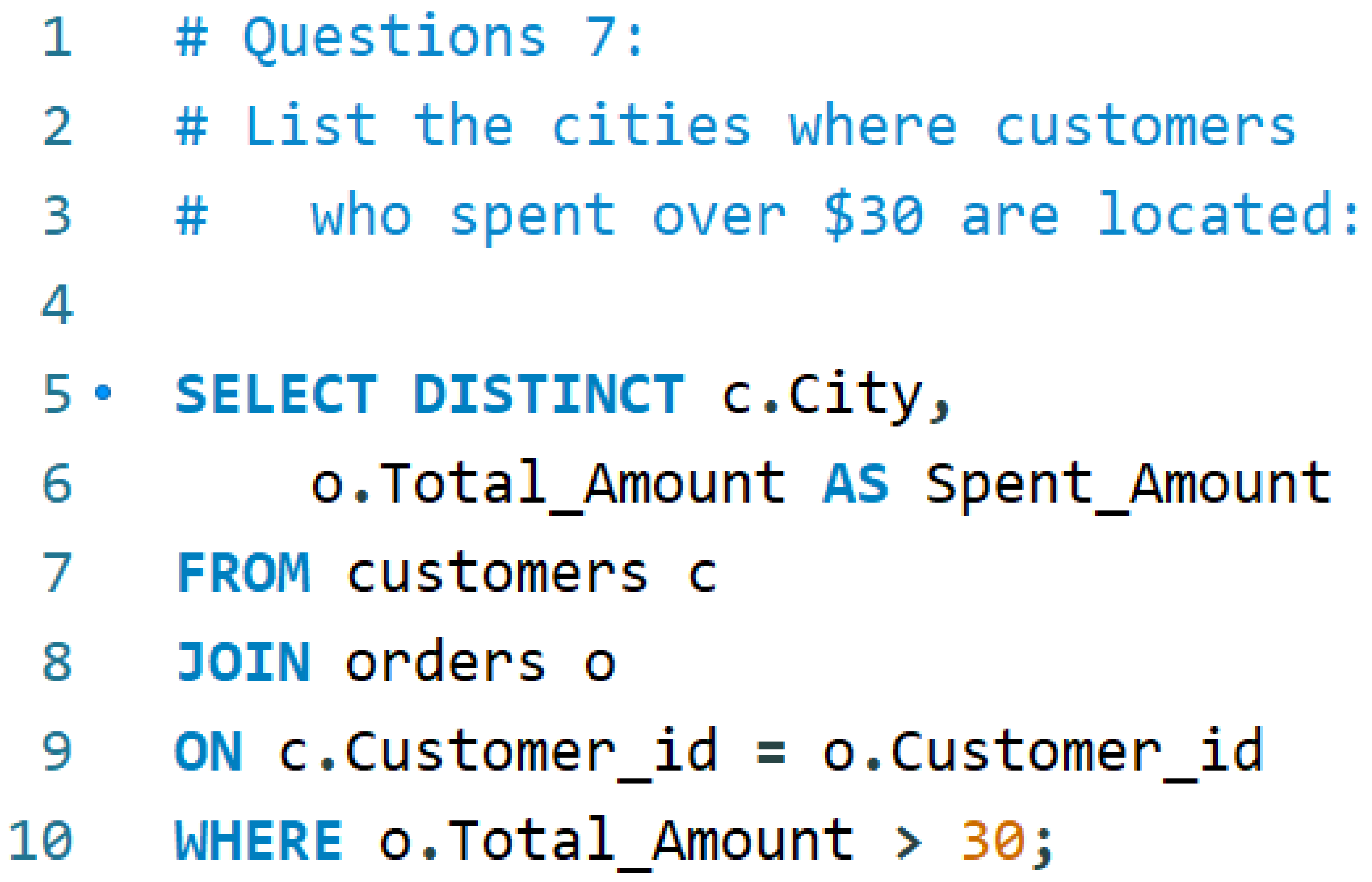
QUESTION 6



```sql
# Questions 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 b.Book_id = o.Book_id
GROUP BY b.Author;
```

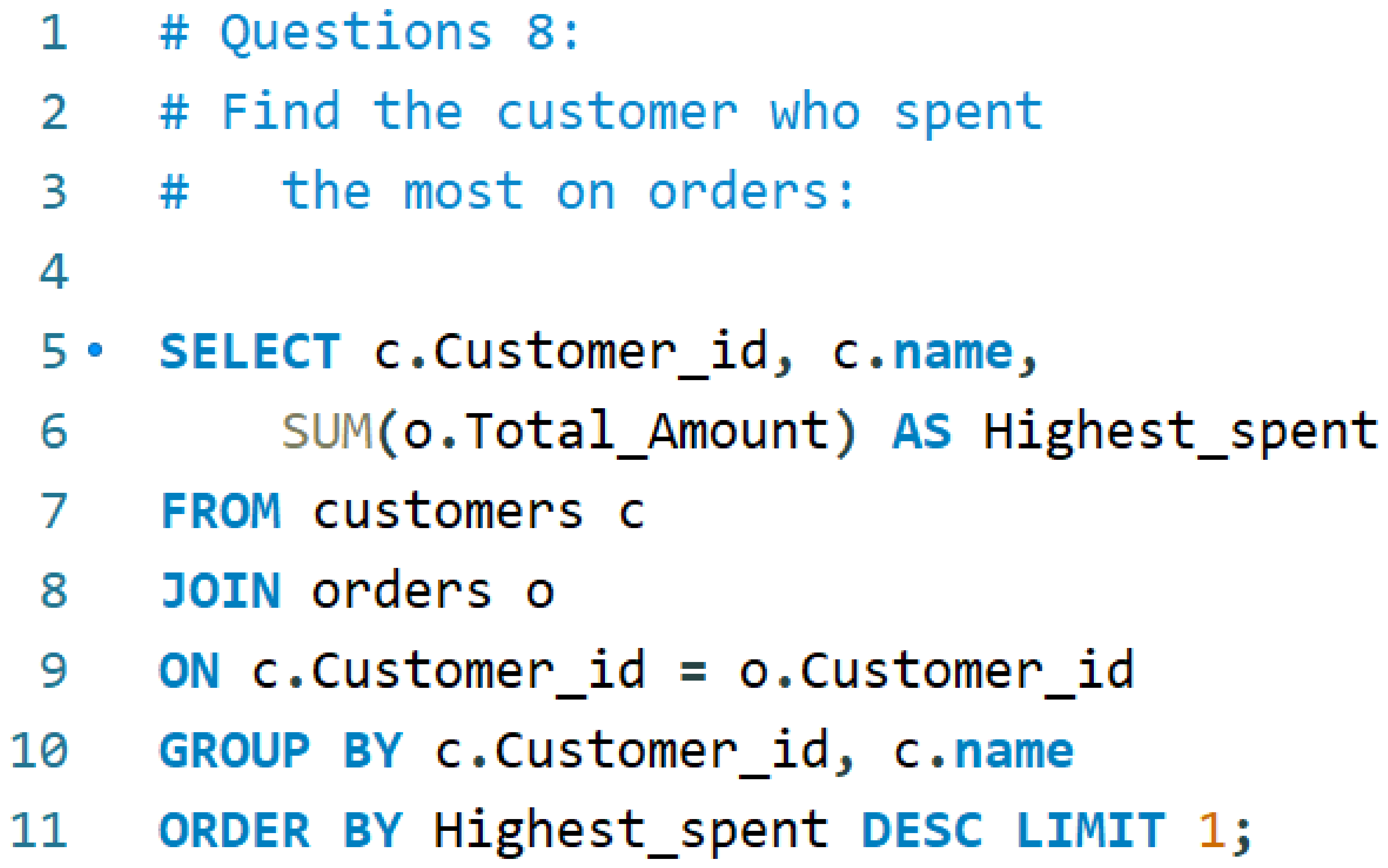
QUESTION 7



```sql
# Questions 7:
# List the cities where customers
#    who spent over $30 are located:

SELECT DISTINCT c.City,
       o.Total_Amount AS Spent_Amount
FROM customers c
JOIN orders o
ON c.Customer_id = o.Customer_id
WHERE o.Total_Amount > 30;
```

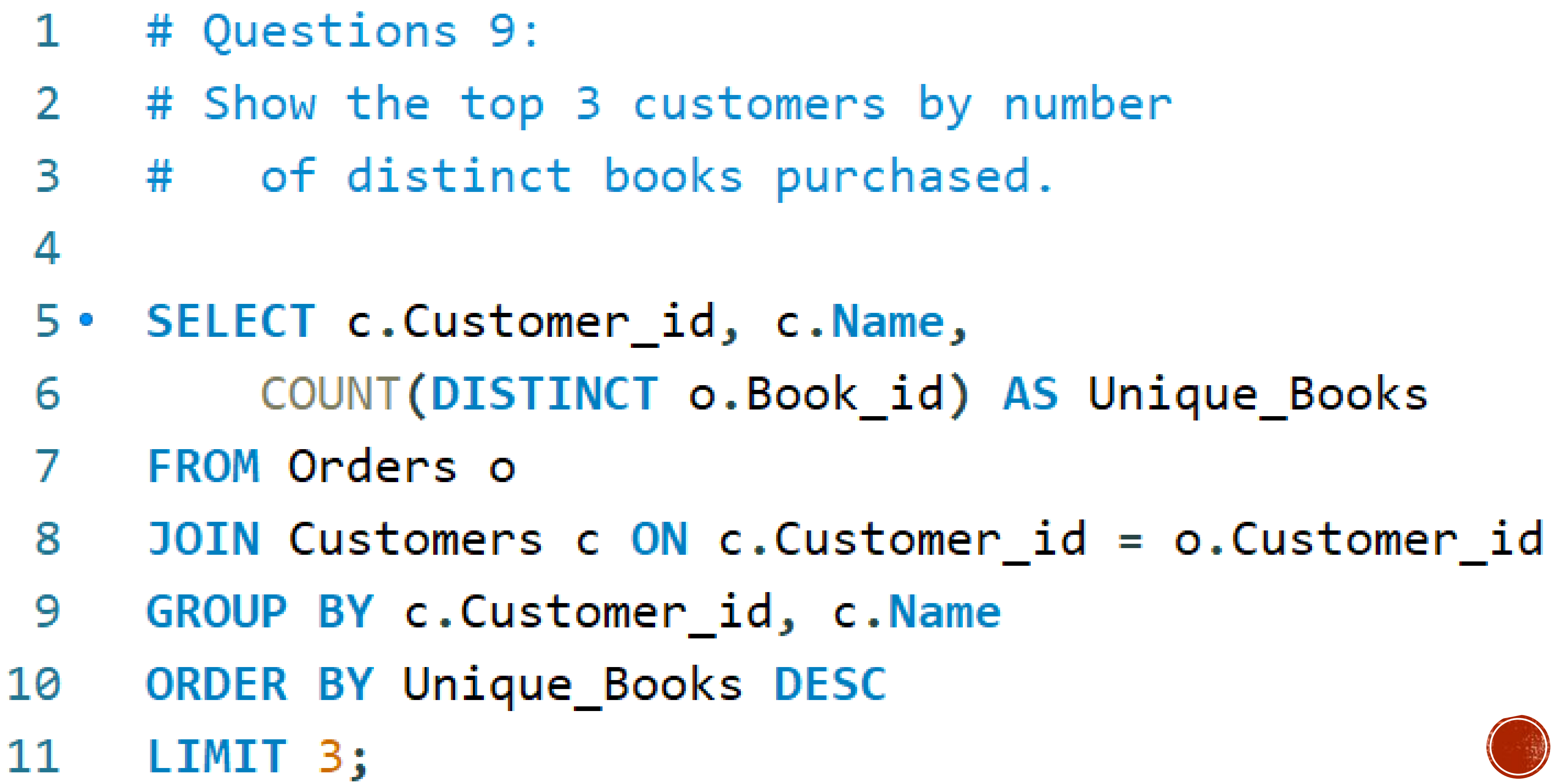
QUESTION 8



```sql
# Questions 8:
# Find the customer who spent
#   the most on orders:

SELECT c.Customer_id, c.name,
    SUM(o.Total_Amount) AS Highest_spent
FROM customers c
JOIN orders o
ON c.Customer_id = o.Customer_id
GROUP BY c.Customer_id, c.name
ORDER BY Highest_spent DESC LIMIT 1;
```

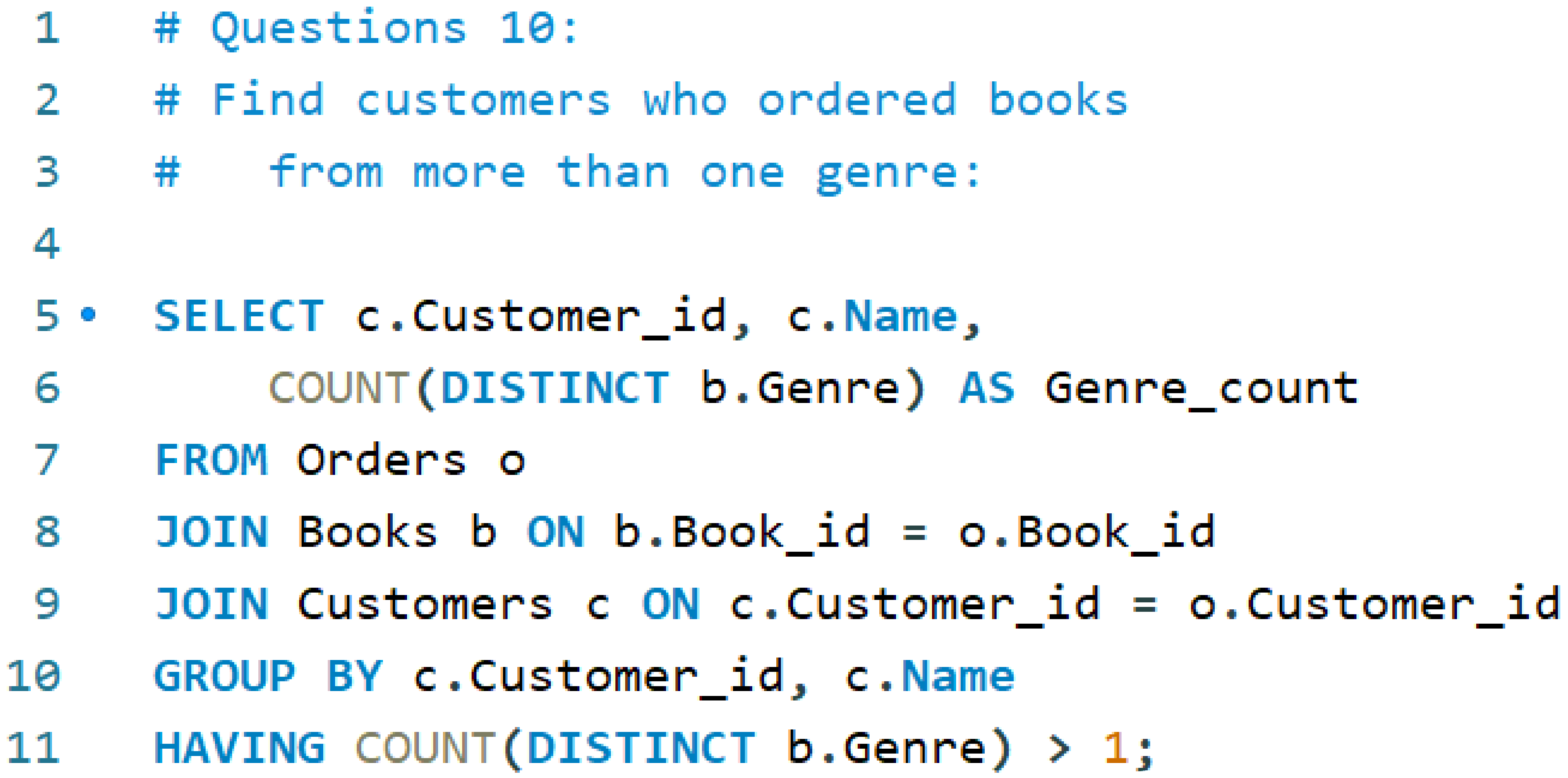
QUESTION 9



```sql
# Questions 9:
# Show the top 3 customers by number
#    of distinct books purchased.

SELECT c.Customer_id, c.Name,
    COUNT(DISTINCT o.Book_id) AS Unique_Books
FROM Orders o
JOIN Customers c ON c.Customer_id = o.Customer_id
GROUP BY c.Customer_id, c.Name
ORDER BY Unique_Books DESC
LIMIT 3;
```

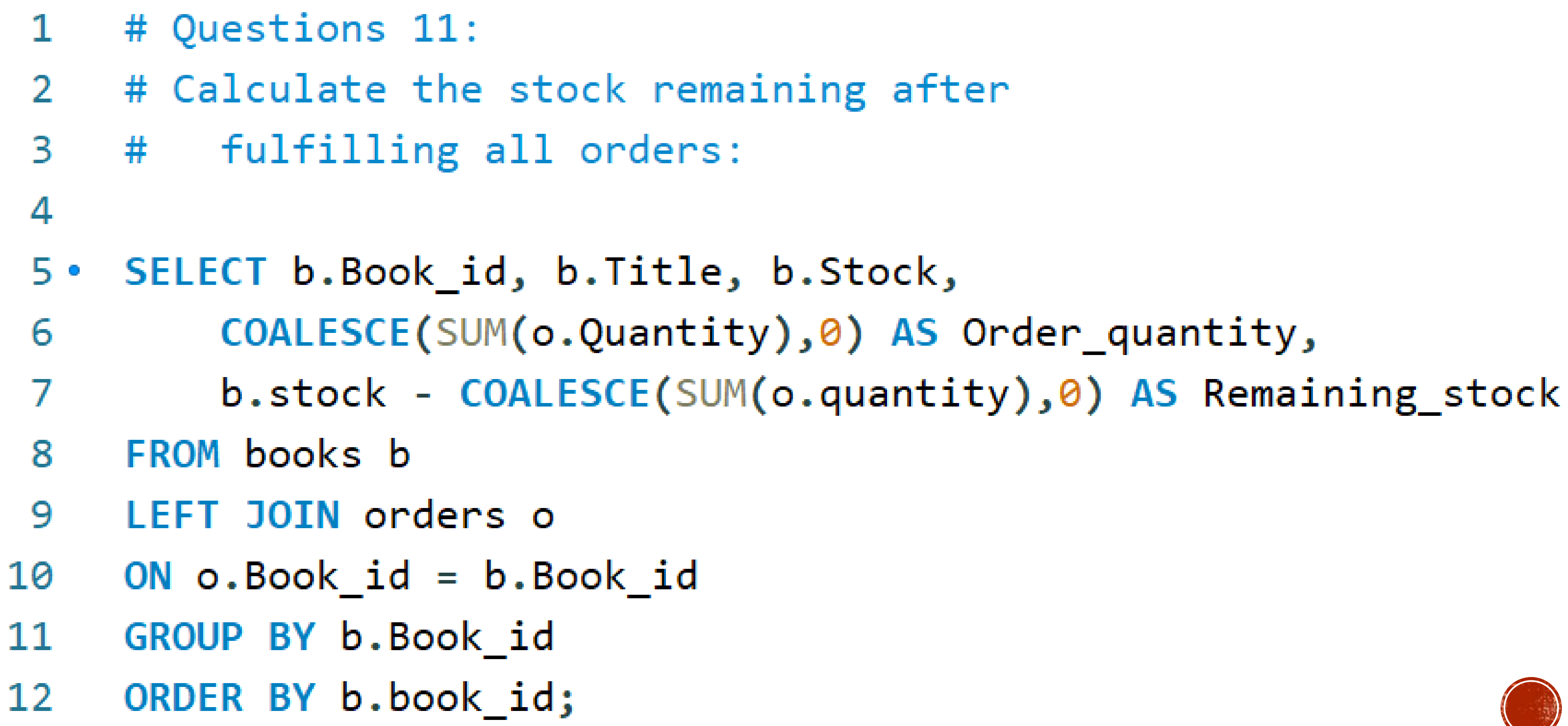
QUESTION 10



```sql
# Questions 10:
# Find customers who ordered books
#    from more than one genre:

SELECT c.Customer_id, c.Name,
    COUNT(DISTINCT b.Genre) AS Genre_count
FROM Orders o
JOIN Books b ON b.Book_id = o.Book_id
JOIN Customers c ON c.Customer_id = o.Customer_id
GROUP BY c.Customer_id, c.Name
HAVING COUNT(DISTINCT b.Genre) > 1;
```

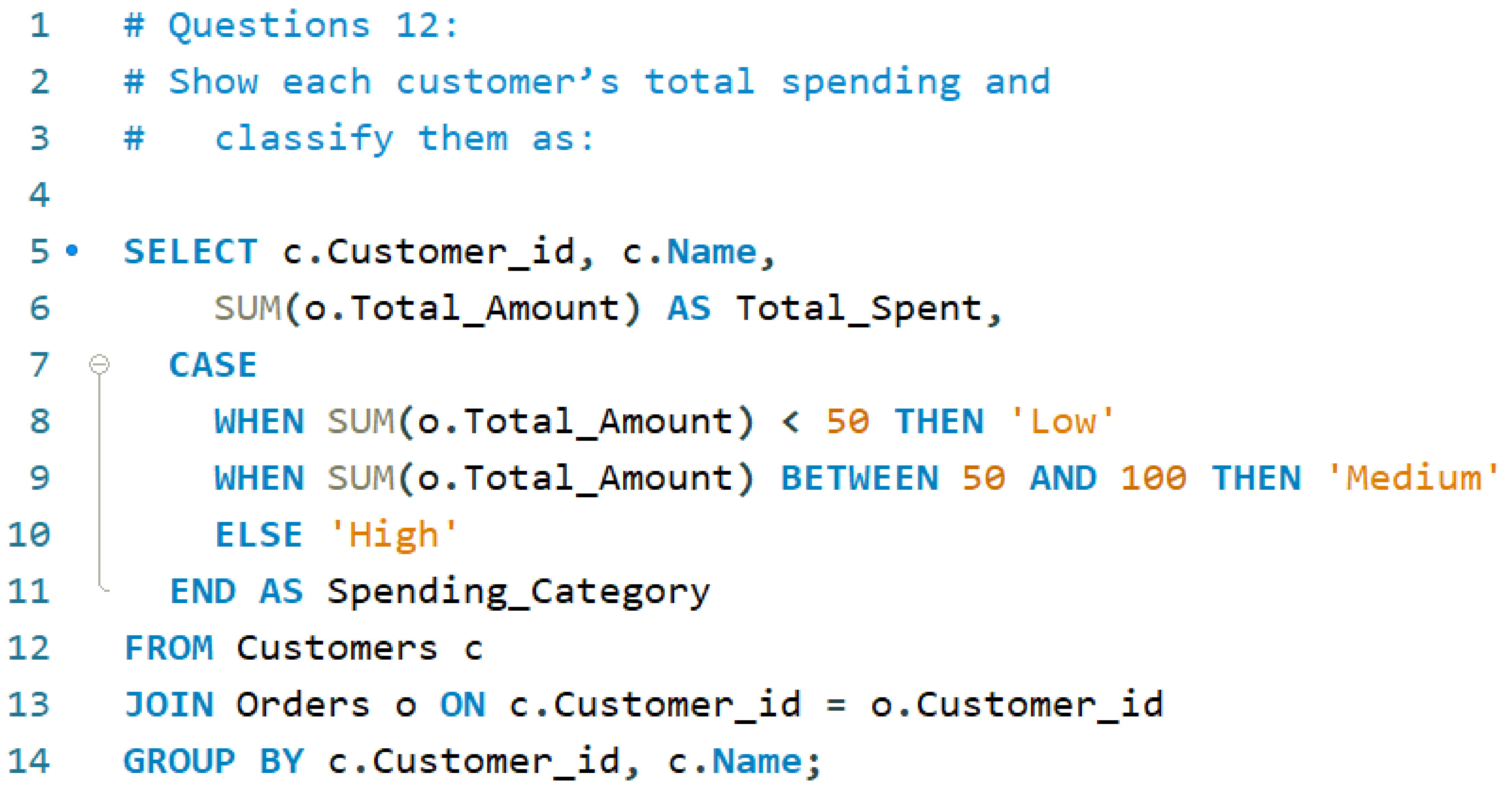
QUESTION 11



```sql
# Questions 11:
# 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_stock
FROM books b
LEFT JOIN orders o
ON o.Book_id = b.Book_id
GROUP BY b.Book_id
ORDER BY b.book_id;
```

QUESTION 12



```sql
# Questions 12:
# Show each customer’s total spending and
#   classify them as:

SELECT c.Customer_id, c.Name,
    SUM(o.Total_Amount) AS Total_Spent,
  CASE
    WHEN SUM(o.Total_Amount) < 50 THEN 'Low'
    WHEN SUM(o.Total_Amount) BETWEEN 50 AND 100 THEN 'Medium'
    ELSE 'High'
  END AS Spending_Category
FROM Customers c
JOIN Orders o ON c.Customer_id = o.Customer_id
GROUP BY c.Customer_id, c.Name;
```