BOOKSTORE DATABASE ANALYSIS

The Data Structure

Books Table

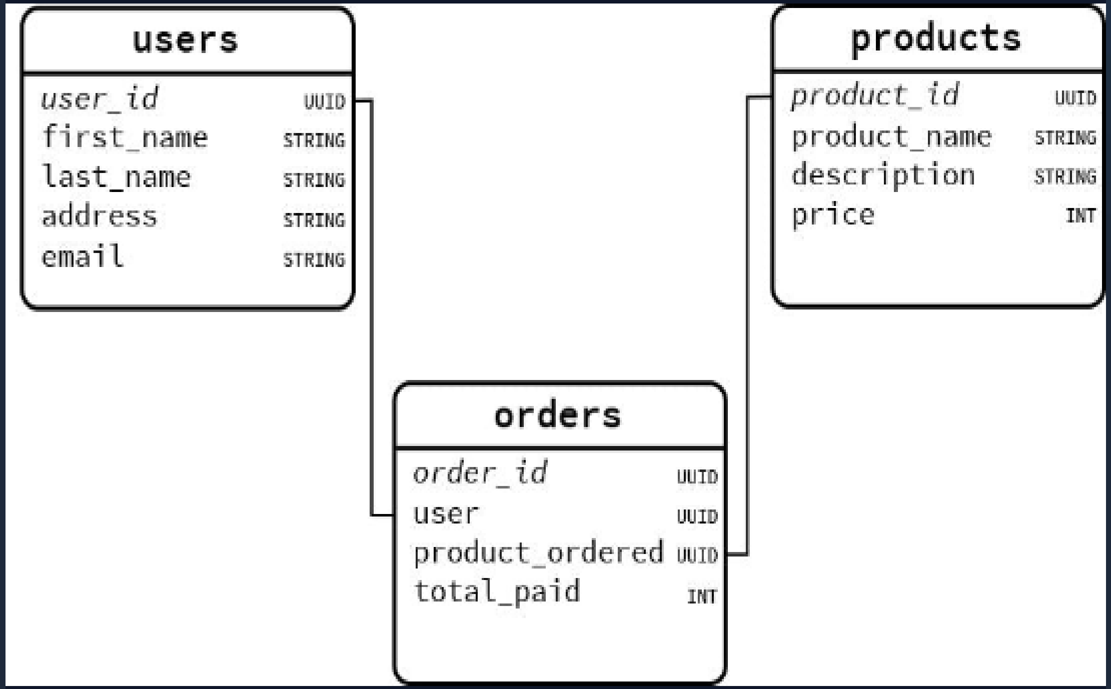
Contains book_id, title, genre, price, stock, published_year, author.

Customers Table

Stores customer_id, name, country, and city.

Orders Table

Links customers and books via order_id, includes quantity, date, and amount.



Basic Filtering

1. Retrieve all "Fiction" books

```
SELECT * FROM Books

WHERE genre='Fiction';
```

2. Books published after 1950

```
SELECT * FROM Books

WHERE published_year > 1950;
```

3. List all genres

```
SELECT DISTINCT genre FROM Books;
```

Customer Queries

4. Customers from Denmark

```
SELECT * FROM Customers

WHERE country='Denmark';
```

5. Cities with High Spending Customers (>$30)

```
SELECT DISTINCT c.city

FROM Customers c

JOIN Orders o ON
c.customer_id=o.customer_id

WHERE o.total_amount > 30;
```

Date & Amount Filtering

6. Orders in November 2023

```
SELECT * FROM Orders

WHERE order_date BETWEEN

'2023-11-01' AND '2023-11-30';
```

7. Large Orders (>$20)

```
SELECT * FROM Orders

WHERE total_amount > 20;
```

Key Metrics

8. Total Book Stock

```
SELECT SUM(stock)

FROM Books;
```

9.Total Revenue

```
SELECT SUM(total_amount)

FROM Orders;
```

Extremes & Averages

10. Most Expensive

```sql
SELECT * FROM Books
ORDER BY price DESC
LIMIT 1;
```

11. Lowest Stock

```sql
SELECT * FROM Books
ORDER BY stock ASC
LIMIT 1;
```

12. Avg Fantasy Price

```sql
SELECT AVG(price)
FROM Books
WHERE genre='Fantasy';
```

Genre Performance

13. Total books sold per Genre

```
      SELECT b.genre, SUM(o.quantity)

      FROM Orders o

      JOIN Books b ON
o.book_id=b.book_id

      GROUP BY b.genre;
```

14. Top 3 Expensive Fantasy Books

```
      SELECT * FROM Books

      WHERE genre='Fantasy'

      ORDER BY price DESC

      LIMIT 3;
```

15. Repeat Customers (>= 2 Orders)

```
SELECT c.name, COUNT(o.order_id)
FROM Orders o
JOIN Customers c ...
GROUP BY c.customer_id, c.name
HAVING COUNT(order_id) ≥ 2;
```

16. Top Spending Customer

```
SELECT c.name, SUM(o.total_amount)
FROM Customers c
JOIN Orders o ...
GROUP BY c.customer_id, c.name
ORDER BY sum DESC LIMIT 1;
```

Product & Author Trends

17. Most Frequently Ordered Book

```
SELECT b.title, COUNT(o.order_id)
FROM Orders o
JOIN Books b ...
GROUP BY b.book_id, b.title
ORDER BY count DESC LIMIT 1;
```

18. Total Quantity Sold by Author

```
SELECT b.author, SUM(o.quantity)
FROM Books b
JOIN Orders o ON b.book_id=o.book_id
GROUP BY b.author;
```

Advanced Inventory

Calculating stock remaining after fulfilling all orders requires a LEFT JOIN and COALESCE to handle books with zero orders.

20. Calculate Remaining Stock

```
SELECT b.title, b.stock,
b.stock - COALESCE(SUM(quantity),0)
AS remaining_quantity
FROM Books b
LEFT JOIN Orders o
ON b.book_id=o.book_id
GROUP BY b.book_id, b.title;
```

Thank you for reviewing the Book Store Database Project.

SQL Project Completed