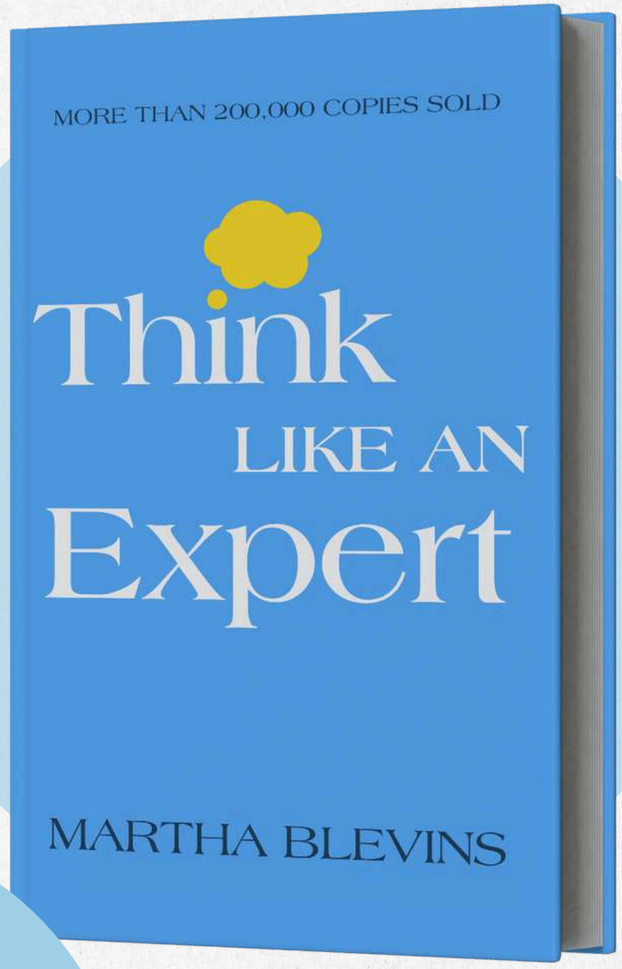
SQL PROJECT

Book Store



ONLINE BOOK STORE

- **Objective**
- To analyze book sales, customer behavior, and revenue trends
- To identify top-selling books, customer preferences, and stock insights

- **Purpose:**
- Analyzing bookstore sales and customer trends

- **By Vivek Choure**

Overview

- **Dataset Description**
- **Books Table:**
- Contains book details such as ID, title, author, genre, published year, price, and stock
- **Customers Table**:
- Includes customer information like name, email, phone, city, and country
- **Orders Table**:
- Tracks customer purchases with order ID, book ID, quantity, order date, and total amount

01 Retrieve all books in the "Fiction" genre:

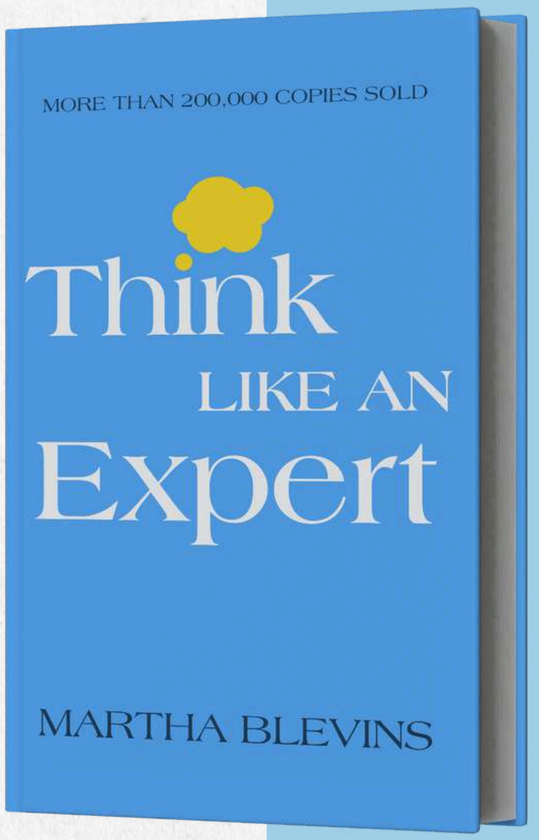
QUERY AND RESULT

```sql
-- 1) Retrieve all books in the "Fiction" genre:

use online_book_store;
SELECT * FROM Books
WHERE Genre='Fiction';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Book_ID	Title	Author	Genre	Published_Year	Price	Stock
4	Customizable 24hour product	Christopher Andrews	Fiction	2020	43.52	8
22	Multi-layered optimizing migration	Wesley Escobar	Fiction	1908	39.23	78
28	Expanded analyzing portal	Lisa Coffey	Fiction	1941	37.51	79
29	Quality-focused multi-tasking challenge	Katrina Underwood	Fiction	1905	31.12	100
31	Implemented encompassing conglomeration	Melissa Taylor	Fiction	2010	21.23	44
39	Optimized national process improvement	Megan Goodwin	Fiction	1978	10.99	42
40	Adaptive didactic interface	Natalie Gonzalez	Fiction	1923	25.97	94
47	Reverse-engineered directional conglomeration	John Christian	Fiction	2006	20.37	90
62	Re-contextualized real-time strategy	Nicole Lynch	Fiction	1953	26.34	23
63	Polarized heuristic database	Franklin Mack	Fiction	1989	22.38	56
100	Synchronized client-server service-desk	James Alvarado	Fiction	1906	49.89	29
116	Multi-tiered foreground contingency	Jamie Gates	Fiction	1938	41.82	50
125	Public-key analyzing Graphic Interface	Abigail Madden	Fiction	1990	32.41	16
130	Realigned context-sensitive pricing structure	Jason Rodriguez	Fiction	2004	6.64	90



02 Find books published after the year 1950:

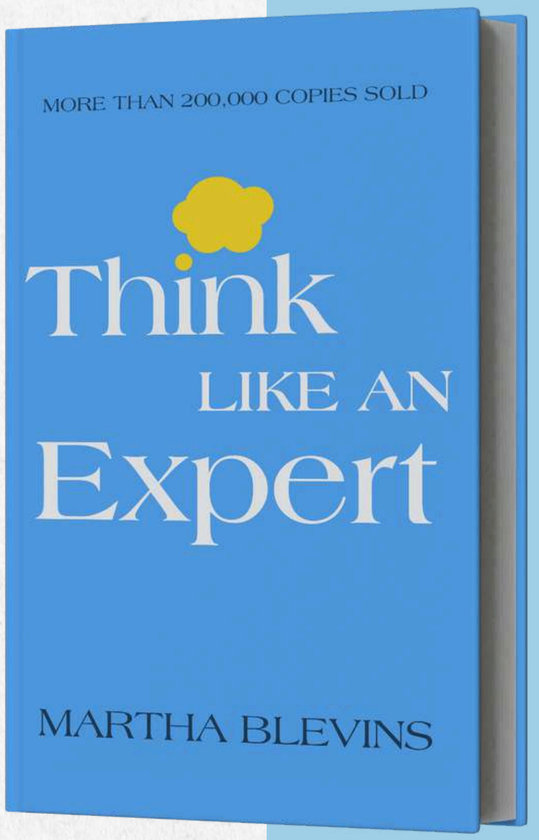
QUERY AND RESULT

```sql
-- 2) Find books published after the year 1950:

SELECT
    *
FROM
    books
WHERE
    Published_Year > 1950
ORDER BY Published_Year;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Book_ID	Title	Author	Genre	Published_Year	Price	Stock
166	Customizable discrete Graphical User Interface	Rebecca Alexander	Romance	1951	11.02	56
174	Pre-emptive executive knowledge user	Rebecca Mann	Mystery	1951	37.83	18
432	Horizontal disintermediate alliance	Rodney Ward	Non-Fiction	1951	8.84	55
43	Function-based zero-defect initiative	Daniel Nunez	Romance	1952	47.39	61
150	Phased logistical open system	Jenna Henderson	Biography	1952	31.95	32
62	Re-contextualized real-time strategy	Nicole Lynch	Fiction	1953	26.34	23
156	Synergistic grid-enabled website	Brandon Black	Fiction	1953	31.68	34
243	Automated systemic toolset	Tiffany Conley	Fantasy	1953	8.87	65
457	Configurable disintermediate extranet	Melissa Lewis	Mystery	1953	28.22	2
167	User-friendly radical standardization	Leon Davis	Science Fiction	1954	36.02	55
193	Customer-focused tertiary methodology	Justin Garcia	Fantasy	1954	29.54	100



04 Show orders placed in November 2023:

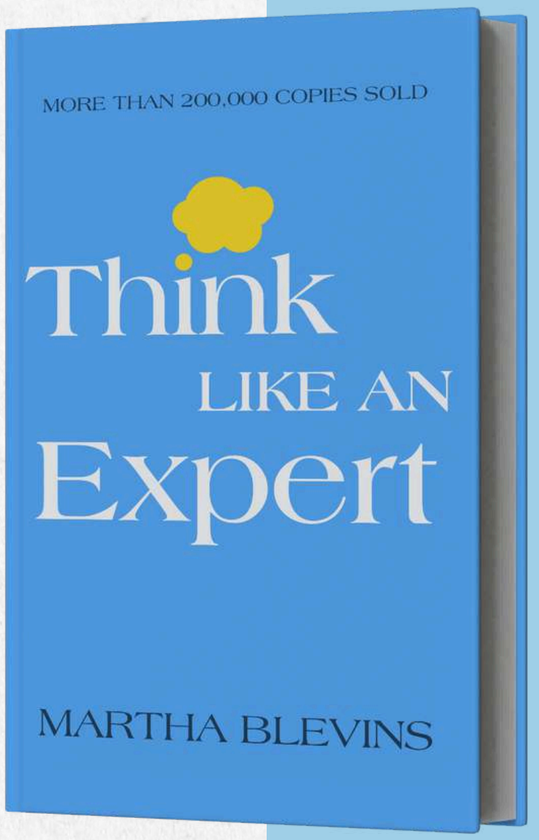
QUERY AND RESULT

```sql
-- 4) Show orders placed in November 2023:

SELECT
    *
FROM
    Orders
WHERE
    Order_Date BETWEEN '2023-11-01' AND '2023-11-30';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Order_ID	Customer_ID	Book_ID	Order_Date	Quantity	Total_Amount
4	433	343	2023-11-25	7	301.21
19	496	60	2023-11-17	9	316.26
75	291	375	2023-11-30	5	170.75
132	469	333	2023-11-22	7	194.32
137	474	471	2023-11-25	8	363.04
163	207	384	2023-11-23	3	101.76
182	129	293	2023-11-01	7	125.51
200	313	303	2023-11-23	1	6.57
213	325	447	2023-11-17	7	253.75
231	22	384	2023-11-11	1	33.92
245	386	97	2023-11-01	9	411.66
252	405	287	2023-11-15	5	237.10



05 Retrieve the total stock of books available:

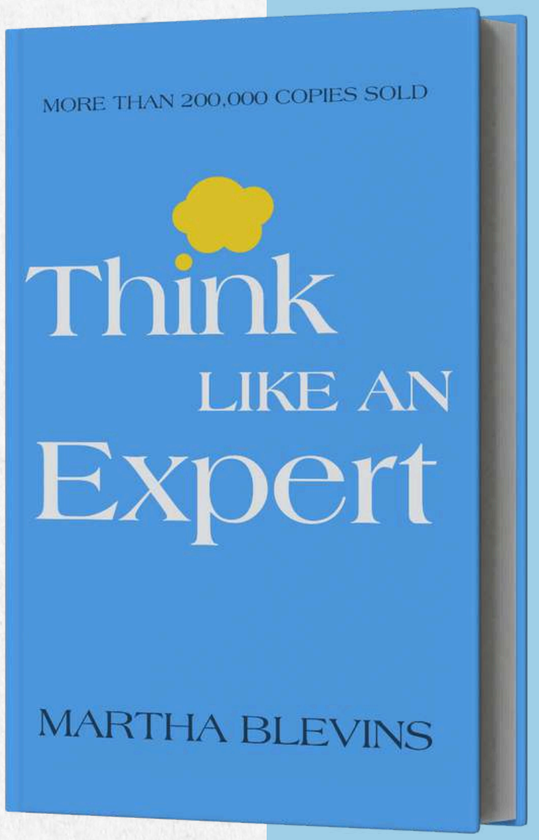
QUERY AND RESULT

```sql
-- 5) Retrieve the total stock of books available:

SELECT
    SUM(Stock) AS total_stocks
FROM
    books;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_stocks
25056



06 Find the details of the most expensive book:

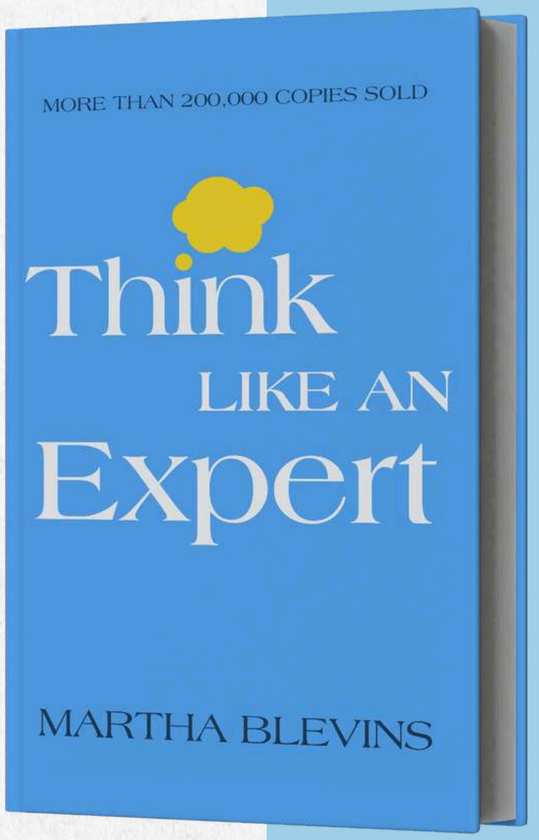
QUERY AND RESULT

```
-- 6) Find the details of the most expensive book:

select * from books
order by Price desc
limit 1;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cont

Book_ID	Title	Author	Genre	Published_Year	Price	Stock
340	Proactive system-worthy orchestration	Robert Scott	Mystery	1907	49.98	88
NULL	NULL	NULL	NULL	NULL	NULL	NULL



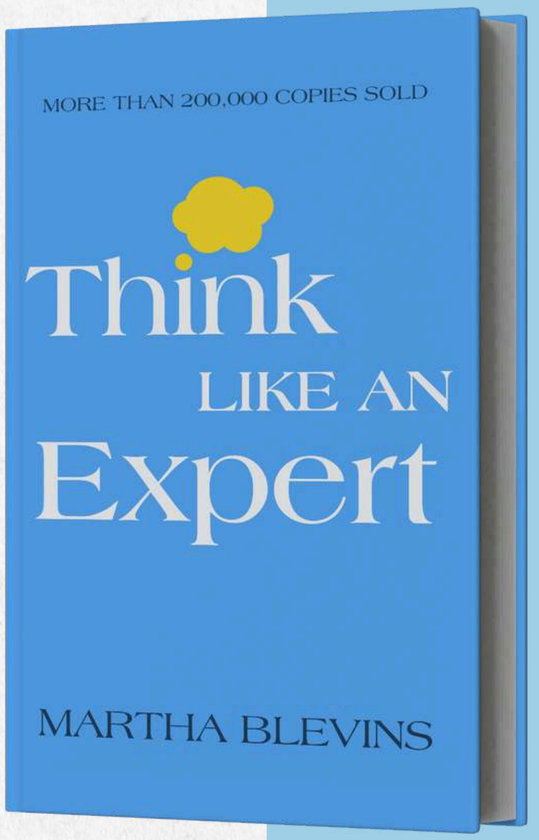
07 Show all customers who ordered more than 1 quantity of a book::

QUERY AND RESULT

```sql
-- 7) Show all customers who ordered more than 1 quantity of a book:
SELECT
    *
FROM
    customers c
        JOIN
    Orders o ON c.Customer_ID = o.Customer_ID
WHERE
    o.Quantity > 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Customer_ID	Name	Email	Phone	City	Country	Order_ID	Customer_ID	Book_ID	Order_Date	Quantity
84	Gary Blair	leonardtaylor@bowers.net	1234567974	Lake Paul	Armenia	1	84	169	2023-05-26	8
137	Steven Miller	tsummers@yahoo.com	1234568027	North Keith	Papua New Guinea	2	137	301	2023-01-23	10
216	Phillip Allen	brianwatkins@gmail.com	1234568106	Kelseyfort	Micronesia	3	216	261	2024-05-27	6
433	Corey Wells	swagner@robertson.net	1234568323	East David	Zambia	4	433	343	2023-11-25	7
14	John Wood	johnsonalexander@gmail.com	1234567904	Richardsonville	New Caledonia	5	14	431	2023-07-26	7
439	Shane Chang	melanie56@fleming.com	1234568329	Ramosstad	Liechtenstein	6	439	119	2024-10-11	5
195	Dominique Turner	tracy39@smith-miller.com	1234568085	Rogersborough	Djibouti	7	195	467	2023-10-23	6
32	Jeffrey Shannon	nmcmillan@smith.com	1234567922	New Carlosbury	Malta	8	32	159	2024-05-07	4
109	Jacob Kelley	qbarber@shelton.org	1234567999	Ravenberg	Macao	9	109	407	2024-01-04	9
94	Mr. David Cox	millermichael@gmail.com	1234567984	West Anthony	New Zealand	10	94	122	2024-07-09	4
454	April Anderson	calexander@yahoo.com	1234568344	Micheleborough	Guyana	12	454	3	2024-06-17	2
420	Andrew Murray	donaldjohnson@yahoo.com	1234568310	North Joseph	Western Sahara	13	420	180	2023-06-08	5



08 Retrieve all orders where the total amount exceeds $20:

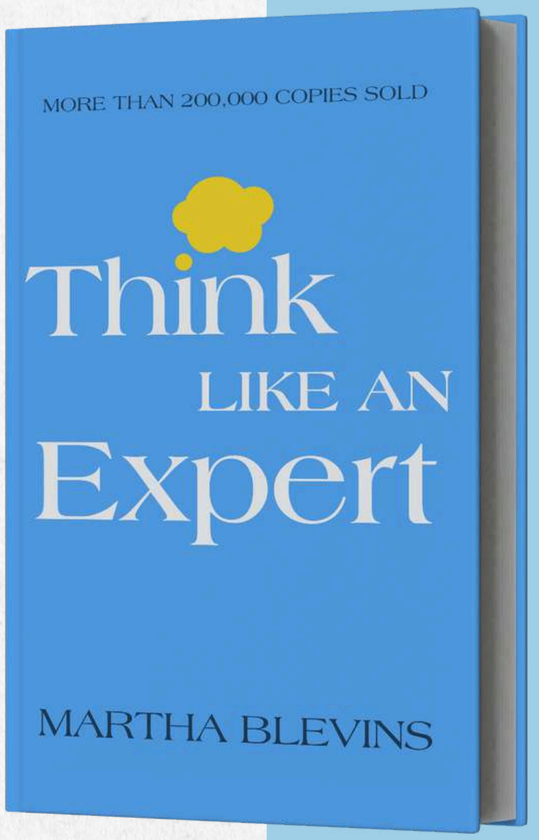
QUERY AND RESULT

```sql
-- 8) Retrieve all orders where the total amount exceeds $20:

SELECT
    *
FROM
    orders
WHERE
    Total_Amount > 20
ORDER BY Total_Amount ASC;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Order_ID	Customer_ID	Book_ID	Order_Date	Quantity	Total_Amount
307	368	133	2023-11-17	1	20.96
100	207	63	2023-07-14	1	22.38
260	112	170	2024-05-07	2	22.56
319	172	296	2024-06-27	2	22.98
345	325	253	2023-07-04	2	23.32
494	117	127	2024-11-23	2	23.32
288	6	128	2023-11-13	1	24.04
188	46	425	2023-03-13	1	24.23
394	299	425	2024-04-14	1	24.23
272	228	321	2024-11-02	1	24.32
415	336	10	2023-07-07	1	24.63



09 List all genres available in the Books table:

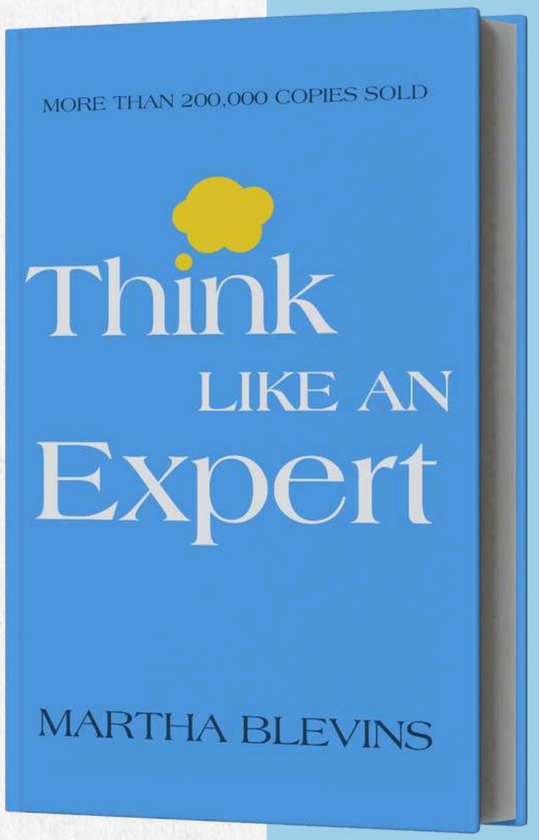
QUERY AND **RESULT**

```sql
-- 9) List all genres available in the Books table:

SELECT DISTINCT
    Genre
FROM
    books;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Genre
Biography
Fantasy
Non-Fiction
Fiction
Romance
Science Fiction
Mystery



10 Find the book with the lowest stock:

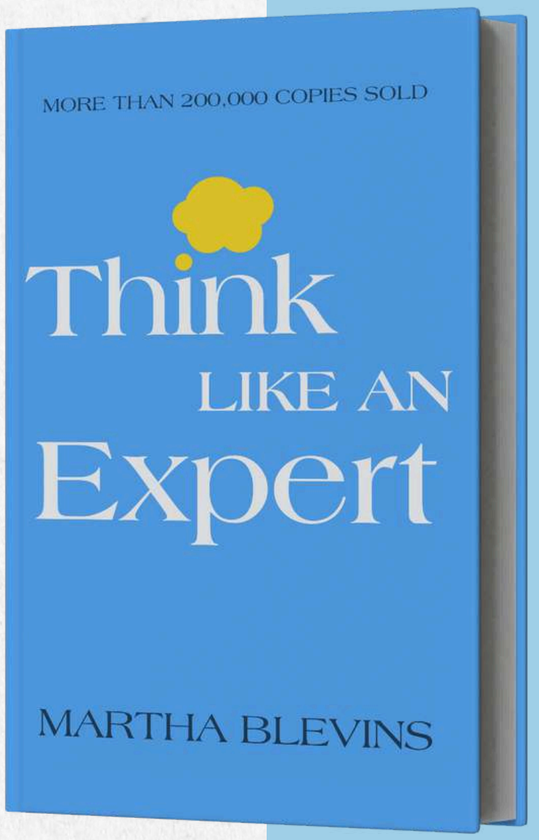
QUERY AND RESULT

```sql
-- 10) Find the book with the lowest stock:

SELECT
    *
FROM
    books
ORDER BY Stock ASC
LIMIT 1;
```

Result Grid

Book_ID	Title	Author	Genre	Published_Year	Price	Stock
44	Networked systemic implementation	Ryan Frank	Science Fiction	1965	13.55	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL



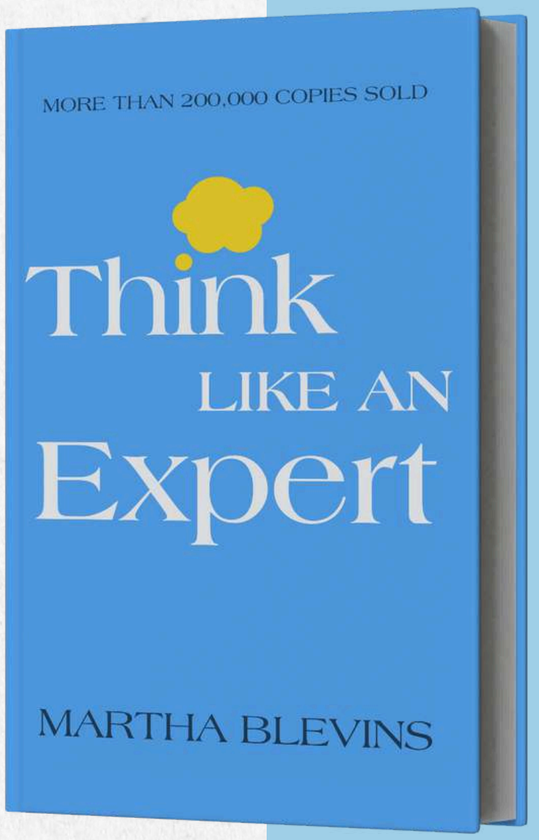
11 Calculate the total revenue generated from all orders:

QUERY AND RESULT

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

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

Revenue
75628.66



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

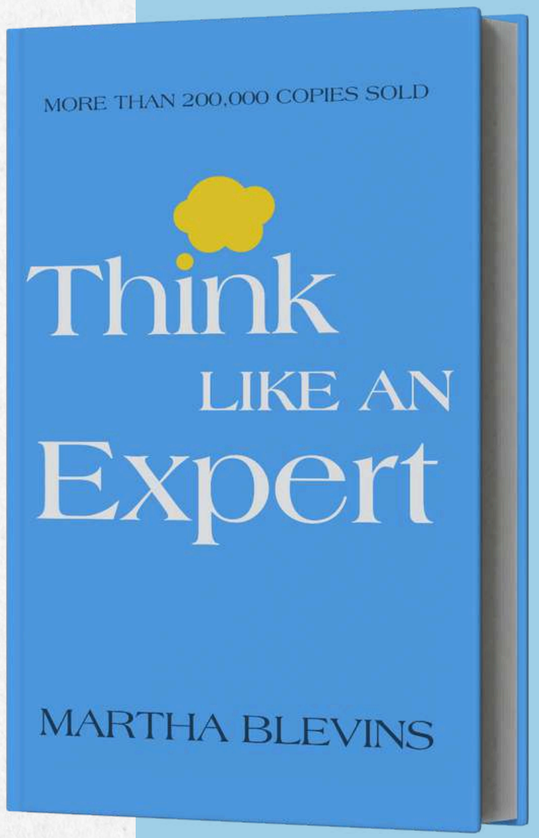
QUERY AND **RESULT**

```sql
-- Advance Questions :
-- 1) Retrieve the total number of books sold for each genre:

SELECT
    Genre, SUM(Quantity) AS Total_books
FROM
    (SELECT
        b.Book_ID, b.Genre, o.Quantity
    FROM
        books b
    JOIN orders o ON b.Book_ID = o.Book_Id) AS temp
GROUP BY Genre;
```

Result Grid | Filter Rows:

Genre	Total_books
Biography	285
Fantasy	446
Science Fiction	447
Mystery	504
Romance	439
Non-Fiction	351
Fiction	225



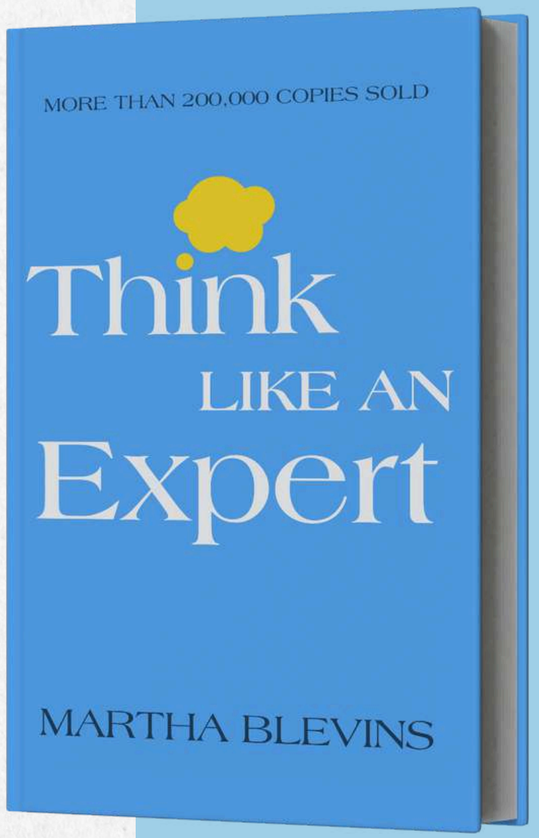
13 Find the average price of books in the "Fantasy" genre:

QUERY AND RESULT

```
-- 2) Find the average price of books in the "Fantasy" genre:
SELECT
    b.Genre, ROUND(AVG(b.price), 2) AS average_price
FROM
    books b
WHERE
    b.Genre = 'Fantasy'
GROUP BY b.Genre;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Genre	average_price
Fantasy	25.98



14 List customers who have placed at least 2 orders:

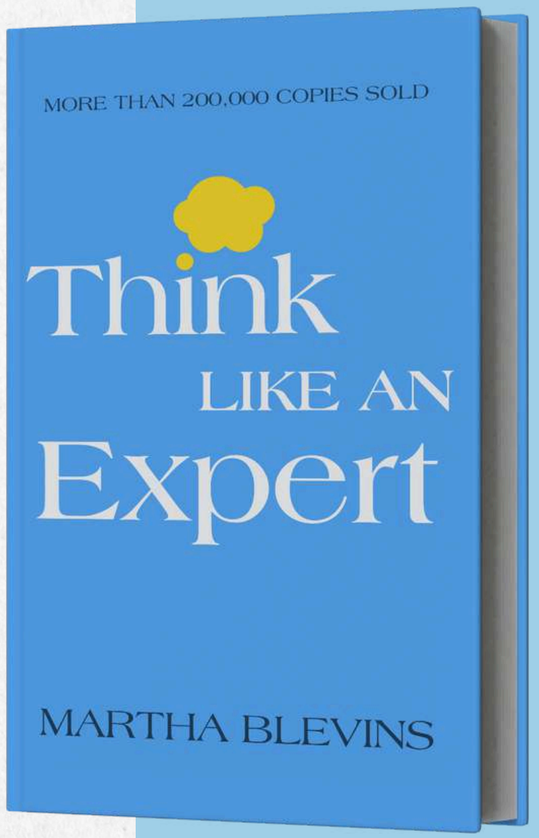
QUERY AND RESULT

```sql
-- 3) List customers who have placed at least 2 orders:

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	name	ORDER_COUNT
84	Gary Blair	2
137	Steven Miller	2
216	Phillip Allen	2
14	John Wood	2
195	Dominique Turner	3



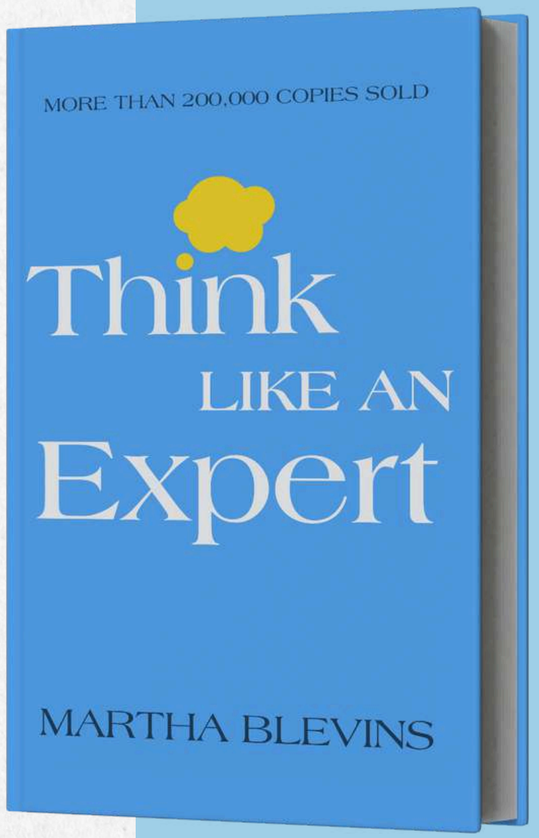
15 Find the most frequently ordered book:

QUERY AND RESULT

```sql
-- 4) Find the most frequently ordered book:
SELECT
    b.Title, COUNT(o.Order_ID) AS number_of_times_book_ordered
FROM
    books b
        JOIN
    orders o ON b.Book_ID = o.Book_ID
GROUP BY b.Title
ORDER BY number_of_times_book_ordered DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Title	number_of_times_book_ordered
Robust tangible hardware	4



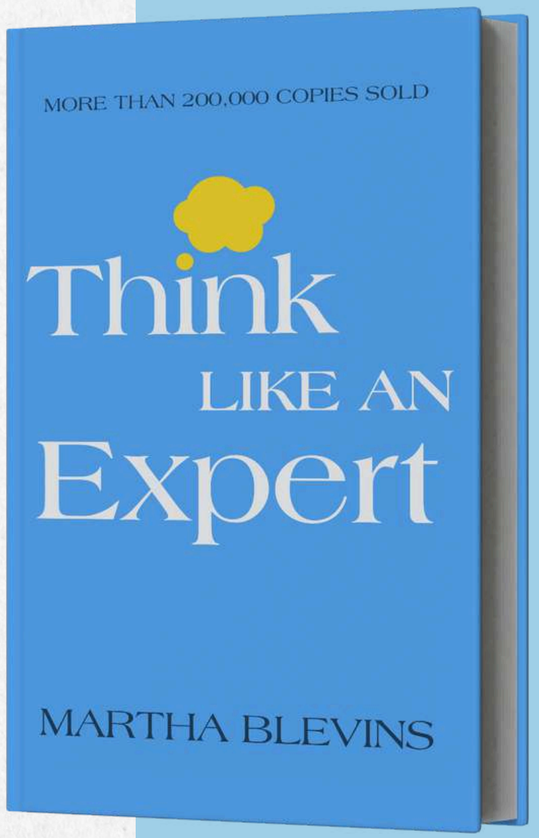
16 Show the top 3 most expensive books of 'Fantasy' Genre :

QUERY AND RESULT

```sql
-- 5) Show the top 3 most expensive books of 'Fantasy' Genre :
use online_book_store;
SELECT
    *
FROM
    books b
WHERE
    b.Genre = 'Fantasy'
ORDER BY b.Price DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch

Book_ID	Title	Author	Genre	Published_Year	Price	Stock
240	Stand-alone content-based hub	Lisa Ellis	Fantasy	1957	49.90	41
462	Innovative 3rdgeneration database	Allison Contreras	Fantasy	1988	49.23	62
238	Optimized even-keeled analyzer	Sherri Griffith	Fantasy	1975	48.97	72
NULL	NULL	NULL	NULL	NULL	NULL	NULL



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

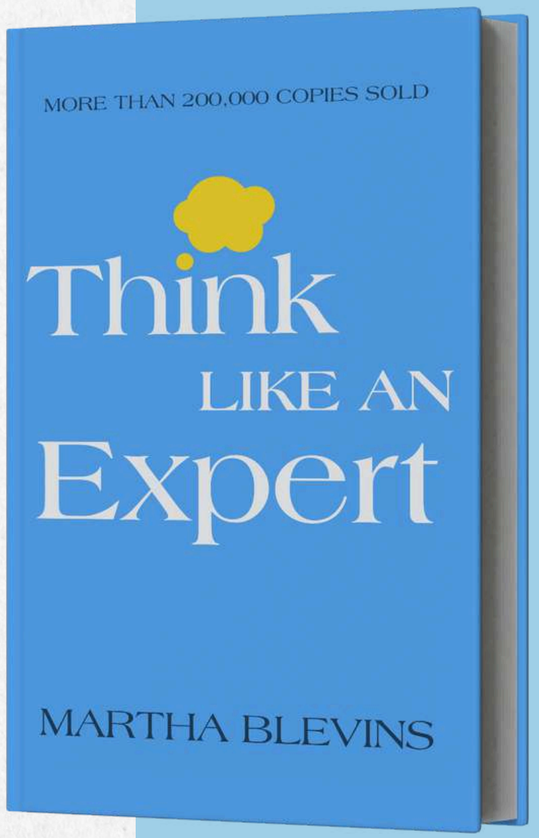
QUERY AND RESULT

```sql
-- 6) Retrieve the total quantity of books sold by each author:

SELECT
    b.Author, SUM(o.Quantity) AS book_sold_by_author
FROM
    books b
        JOIN
    orders o ON b.Book_ID = o.Book_ID
GROUP BY b.Author
ORDER BY book_sold_by_author DESC;
```

Result Grid | Filter Rows:

Author	book_sold_by_author
Patrick Contreras	28
Melissa Taylor	27
Emily James	24
Thomas Trujillo	24
Erica Parker	23
Sheena Harris	23
Valerie Moore	23
Ellen Doyle	23
Rachel Gibbs	22
Amanda Wilson	22
Tonya Saunders	21
Kristi Phillips	21
Anna Roberts	21



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

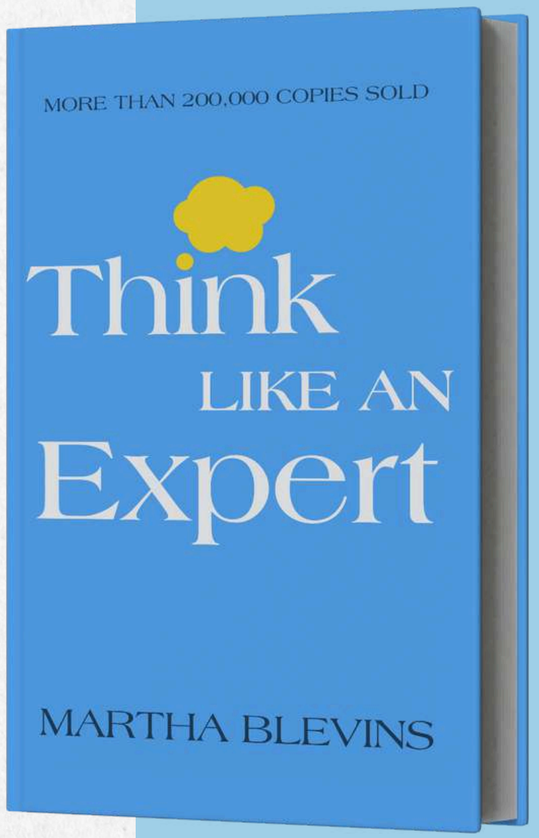
QUERY AND RESULT

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

SELECT DISTINCT
    c.city, total_amount
FROM
    orders o
        JOIN
    customers c ON o.customer_id = c.customer_id
WHERE
    o.total_amount > 30;
```

Result Grid | Filter Rows:

city	total_amount
Lake Paul	188.56
North Keith	216.60
Kelseyfort	85.50
East David	301.21
Richardsonville	136.36
Ramosstad	249.40
Rogersborough	82.92
New Carlosbury	144.84
Ravenberg	379.71
West Anthony	123.00
North Carolyn	38.01
Michelebrough	31.50



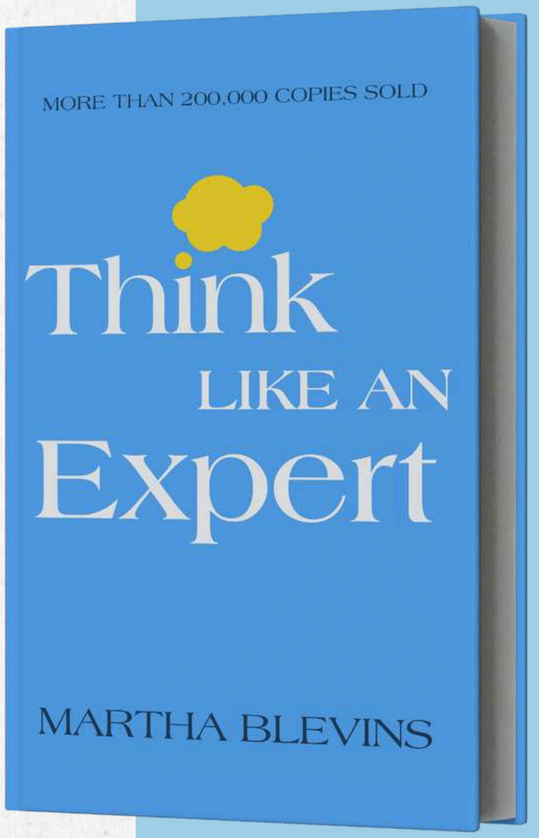
19 Find the customer who spent the most on orders:

QUERY AND RESULT

```
-- 8) Find the customer who spent the most on orders:
SELECT
    c.Customer_ID,
    c.Name,
    c.Phone,
    SUM(o.Total_Amount) AS tatal_spent_amount
FROM
    customers c
        JOIN
    orders o ON c.Customer_ID = o.Customer_ID
GROUP BY c.Customer_ID
ORDER BY tatal_spent_amount DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export:

Customer_ID	Name	Phone	tatal_spent_amount
457	Kim Turner	1234568347	1398.90



20 Calculate the stock remaining after fulfilling all orders:

QUERY AND RESULT

```sql
-- 9) Calculate the stock remaining after fulfilling all orders:
SELECT
    Book_ID,
    Title,
    (stock - total_quantity_ordered) AS remaining_stock
FROM
    (SELECT
        b.Book_ID,
            b.Title,
            b.stock,
            SUM(o.Quantity) AS total_quantity_ordered
    FROM
        books b
    JOIN orders o ON b.Book_ID = o.Book_ID
    GROUP BY b.Book_ID
    ORDER BY b.Book_ID) AS temp
```

Result Grid | Filter Rows: | Export: | Wrap Cell Cont

Book_ID	Title	remaining_stock
1	Configurable modular throughput	97
3	Streamlined coherent initiative	22
5	Adaptive 5thgeneration encoding	8
7	Open-architected exuding structure	90
8	Persistent local encoding	81
10	Ergonomic national hub	24
11	Secured zero tolerance time-frame	5
13	Adaptive 5thgeneration orchestration	90
16	Vision-oriented tangible project	7
17	Reduced secondary core	36
18	Adaptive 4thgeneration concept	18