SQL Bookstore Analysis Project

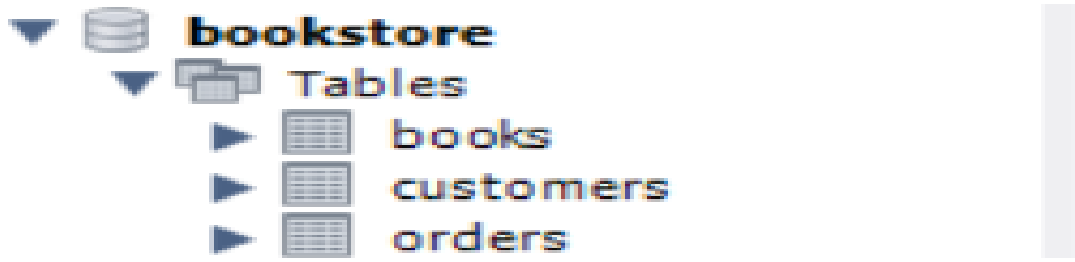
```sql
Create Database bookstore;
use bookstore;

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,
    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,
    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
);
```

Here I am creating one bookstore name database. And in this database I am adding 4 tables – books, customers, and order



- Below are the 20 questions that I analyze through SQL.

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

```sql
-- 2) Find books published after the year 1950
SELECT
    *
FROM
    Books
WHERE
    Published_year > 1950;
```

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

Book_ID	Title	Author	Genre	Published_Year	Price	Stock
2	Persevering reciprocal knowledge user	Mario Moore	Fantasy	1971	35.8	19
4	Customizable 24hour product	Christopher Andrews	Fiction	2020	43.52	8
5	Adaptive 5thgeneration encoding	Juan Miller	Fantasy	1956	10.95	16
6	Advanced encompassing implementation	Bryan Morgan	Biography	1985	6.56	2
8	Persistent local encoding	Troy Cox	Science Fiction	2019	48.99	84
9	Optimized interactive challenge	Colin Buckley	Fantasy	1987	14.33	70
10	Ergonomic national hub	Samantha Ruiz	Mystery	2015	24.63	25
11	Secured zero tolerance time-frame	Denise Barnes	Fantasy	1998	35.95	10
12	Polarized optimal array	Destiny Scott	Non-Fiction	1989	27.43	63
15	User-friendly motivating strategy	Keith Smith	Non-Fiction	1997	23.83	58

```sql
-- 3) List all customers from the Canada:
SELECT
    *
FROM
    Customers
WHERE
    country = 'Canada';
```

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

Customer_ID	Name	Email	Phone	City	Country
38	Nicholas Harris	christine93@perkins.com	1234567928	Davistown	Canada
415	James Ramirez	robert54@hall.com	1234568305	Maxwelltown	Canada
468	David Hart	stokesrebecca@gmail.com	1234568358	Thompsonfurt	Canada
NULL	NULL	NULL	NULL	NULL	NULL

```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

```sql
-- 5) Retrieve the total stock of books available:
SELECT
    SUM(stock) AS Total_Stock
FROM
    Books;
```

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

Total_Stock
25056

```sql
-- 6) Find the most expensive book:
SELECT
    *
FROM
    Books
ORDER BY Price DESC
LIMIT 1;
```

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

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

```sql
-- 7) Show all customers who ordered more than 1 quantity of a book:
SELECT * FROM Orders
WHERE quantity>1;
```

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

Order_ID	Customer_ID	Book_ID	Order_Date	Quantity	Total_Amount
1	84	169	2023-05-26	8	188.56
2	137	301	2023-01-23	10	216.6
3	216	261	2024-05-27	6	85.5
4	433	343	2023-11-25	7	301.21
5	14	431	2023-07-26	7	136.36
6	439	119	2024-10-11	5	249.4
7	195	467	2023-10-23	6	82.92
8	32	159	2024-05-07	4	144.84
9	109	407	2024-01-04	9	379.71
10	94	122	2024-07-09	4	123

```sql
-- 8) Retrieve all orders where the total amount exceeds 100:
SELECT
    *
FROM
    Orders
WHERE
    total_amount > 100;
```

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

Order_ID	Customer_ID	Book_ID	Order_Date	Quantity	Total_Amount
1	84	169	2023-05-26	8	188.56
2	137	301	2023-01-23	10	216.6
4	433	343	2023-11-25	7	301.21
5	14	431	2023-07-26	7	136.36
6	439	119	2024-10-11	5	249.4
8	32	159	2024-05-07	4	144.84
9	109	407	2024-01-04	9	379.71
10	94	122	2024-07-09	4	123
13	420	180	2023-06-08	5	125.45
15	127	479	2023-01-10	6	229.62

```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

```sql
-- 10) Find the book with the lowest stock:
SELECT
    *
FROM
    Books
ORDER BY stock
LIMIT 1;
```

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

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

```sql
-- 11) Calculate the total revenue generated from all orders:
SELECT
    ROUND(SUM(total_amount), 0) AS Revenue
FROM
    Orders;
```

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

Revenue
75629

```sql
-- 12) Retrieve the total number of books sold for each genre:
SELECT
    *
FROM
    ORDERS;
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;
```

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

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

```sql
-- 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';
```

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

Average_Price
28.77

```sql
-- 14) List customers who have placed at least 3 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) >= 3;
```

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

customer_id	name	ORDER_COUNT
195	Dominique Turner	3
94	Mr. David Cox	3
420	Andrew Murray	3
462	James Brewer	3
119	Alyssa Cuevas	3
265	Cassandra Cole	3
386	Pamela Gordon	3
463	Brandon Dunn	3
418	Kiara Blankenship MD	3
415	James Ramirez	3

```sql
-- 15) Find the top 5 most frequently ordered books:
SELECT
    o.Book_id, b.title, COUNT(o.order_id) AS ORDER_COUNT
FROM
    orders o
JOIN
    books b ON o.book_id = b.book_id
GROUP BY o.book_id , b.title
ORDER BY ORDER_COUNT DESC
LIMIT 5;
```

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

Book_id	title	ORDER_COUNT
333	Advanced responsive extranet	4
491	Pre-emptive intangible adapter	4
120	Integrated secondary access	4
88	Robust tangible hardware	4
31	Implemented encompassing conglomeration	4

```sql
-- 16) Show the top 5 most expensive books of 'Mystery' Genre :
SELECT
    *
FROM
    books
WHERE
    genre = 'Mystery'
ORDER BY price DESC
LIMIT 5;
```

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

Book_ID	Title	Author	Genre	Published_Year	Price	Stock
340	Proactive system-worthy orchestration	Robert Scott	Mystery	1907	49.98	88
162	Centralized maximized database	Tracy Pace	Mystery	1981	48.84	64
209	Distributed modular capability	Nicole Berger	Mystery	1979	48.67	63
101	Multi-tiered context-sensitive hub	Amanda Knight	Mystery	1923	48.49	40
42	Pre-emptive interactive focus group	Shannon Reese	Mystery	1937	48.35	82
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```sql
-- 17) 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 o.book_id = b.book_id
GROUP BY b.Author;
```

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

author	Total_Books_Sold
Margaret Moore	8
John Davidson	13
Christopher Fuentes	6
Marissa Smith	16
Christopher Dixon	15
Tonya Saunders	21
Larry Hunt	6
Brandon Foster	4
Michelle Bell	11
Mary French	14

```sql
-- 18) List the cities where customers who spent over 50 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 > 50;
```

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

city	total_amount
Lake Paul	188.56
North Keith	216.6
Kelseyfort	85.5
East David	301.21
Richardsonville	136.36
Ramosstad	249.4
Rogersborough	82.92
New Carlosbury	144.84
Ravenberg	379.71
West Anthony	123

```sql
-- 19) 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
    customers c ON o.customer_id = c.customer_id
GROUP BY c.customer_id , c.name
ORDER BY Total_spent DESC
LIMIT 5;
```

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

customer_id	name	Total_Spent
457	Kim Turner	1398.9
174	Jonathon Strickland	1080.9499999999998
364	Carrie Perez	1052.27
405	Julie Smith	991
386	Pamela Gordon	986.3000000000001

```sql
-- 20) 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_Quantity
FROM
    books b
LEFT JOIN
    orders o ON b.book_id = o.book_id
GROUP BY b.book_id
ORDER BY b.book_id;
```

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

customer_id	name	Total_Spent
457	Kim Turner	1398.9
174	Jonathon Strickland	1080.9499999999998
364	Carrie Perez	1052.27
405	Julie Smith	991
386	Pamela Gordon	986.3000000000001