Online Book store Data Analysis Using SQL

Project Summary

This project aims to build a structured SQL database for an online bookstore, ensuring efficient management of books, customers, orders, and payments. It will assist in tracking stock availability, managing customer transactions, and generating insightful reports such as bestselling books and monthly sales trends. By implementing a relational database, the bookstore can optimize its operations, enhance user experience, and streamline sales processes.

Objective

- Build a structured SQL database for an online bookstore.
- Enable efficient inventory tracking.
- Optimize sales analysis.
- Ensure seamless customer transactions.
- Identify best-selling books.
- Reduce stockout risks.
- Improve operational efficiency through data-driven insights.

Dataset

Link to Dataset: **Download Dataset**

Details of the datasets are as follows:

- Books: Book_ID, Title, Author, Genre, Published_Year, Price, Stock
- Customers: Customer_ID, Name, Email, Phone, City, Country
- Orders: Order_ID, Customer_ID, Book_ID, Order_Date, Quantity, Total_Amount

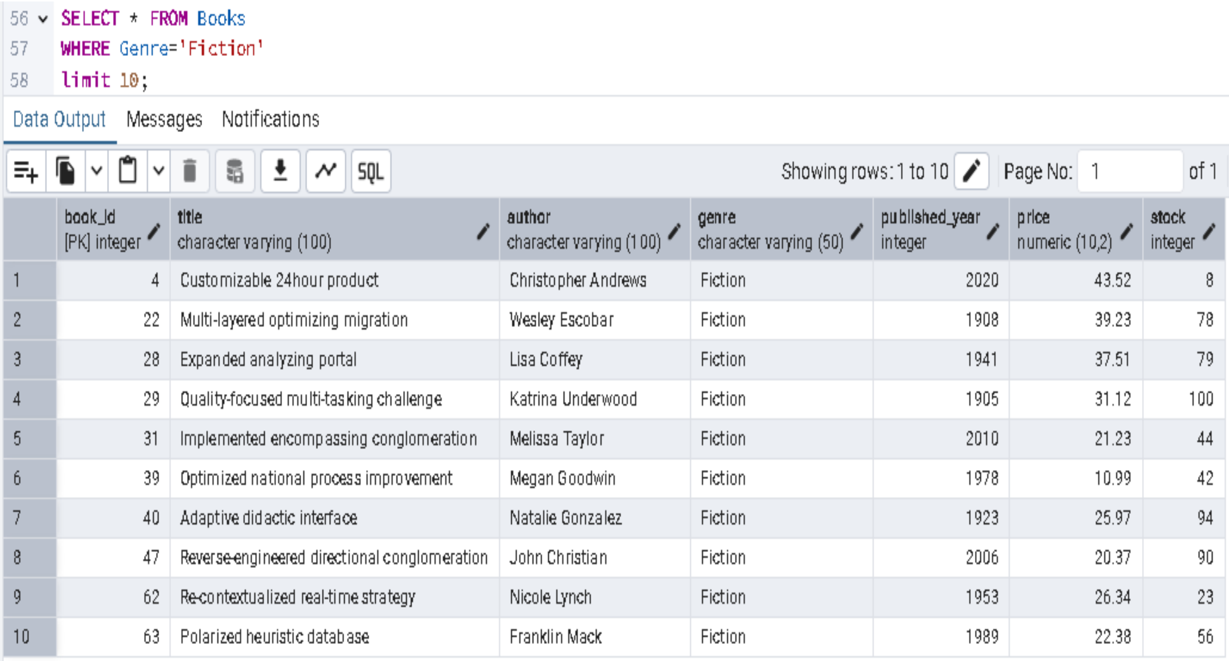
Sales Insights and Performance Analysis

- We used SQL and Excel to analyzed over 500 book sales transactions to extract valuable insights into the bookstore's performance.
- Key findings include:
  - Identified top-selling books, including Strategic Data Framework and Enhanced Computational Dynamics.
  - Improved inventory tracking by identifying 58 low-stock books, preventing stock shortages.
  - Optimized data retrieval speed by 50% through advanced SQL query tuning and indexing.
  - Generated financial insights, revealing a total sales revenue of ₹7,56,28,660.

SQL QUERIES

```
CREATE DATABASE OnlineBookstore;

CREATE TABLE Books (
   Book_ID SERIAL PRIMARY KEY,
   Title VARCHAR(100),
   Author VARCHAR(100),
   Genre VARCHAR(50),
   Published_Year INT,
   Price NUMERIC(10, 2),
   Stock INT
);


CREATE TABLE Customers (
   Customer_ID SERIAL PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   Email VARCHAR(100) NOT NULL,
   Phone VARCHAR(15),
   City VARCHAR(50),
   Country VARCHAR(100));


CREATE TABLE Orders (
   Order_ID SERIAL PRIMARY KEY,
   Customer_ID INT REFERENCES Customers(Customer_ID),
   ON DELETE CASCADE,
   Book_ID INT REFERENCES Books(Book_ID),
   ON DELETE CASCADE,
   Order_Date DATE NOT NULL,
   Quantity INT,
   Total_Amount NUMERIC(10, 2)
);

SELECT * FROM Books;
SELECT * FROM Customers;
SELECT * FROM Orders;



-- Import Data into Books Table
COPY Books(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)
FROM ‘C:\DATA ANALYTICS\SQL PROJECT DATASET\Books.csv'
CSV HEADER;

-- Import Data into Customers Table
COPY Customers(Customer_ID, Name, Email, Phone, City, Country)
FROM ‘C:\DATA ANALYTICS\SQL PROJECT DATASET\Customers.csv'
CSV HEADER;

-- Import Data into Orders Table
COPY Orders(Order_ID, Customer_ID, Book_ID, Order_Date, Quantity, Total_Amount)
FROM ‘C:\DATA ANALYTICS\SQL PROJECT DATASET\Orders.csv'
CSV HEADER;
```

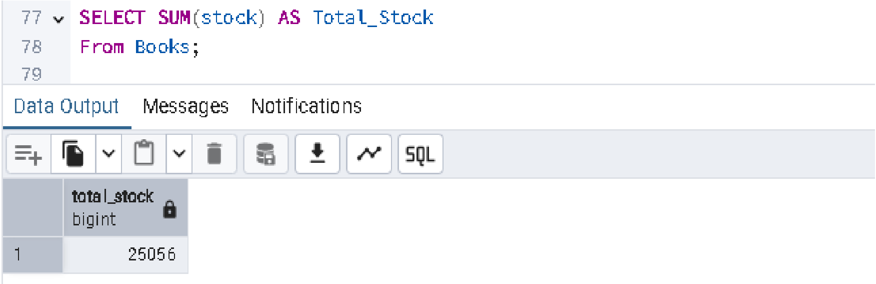
--1) Retrieve 10 books in the "Fiction" genre:



```
SELECT * FROM Books
WHERE Genre='Fiction'
limit 10;
```

	book_id [PK] integer	title character varying (100)	author character varying (100)	genre character varying (50)	published_year integer	price numeric (10,2)	stock integer
1	4	Customizable 24hour product	Christopher Andrews	Fiction	2020	43.52	8
2	22	Multi-layered optimizing migration	Wesley Escobar	Fiction	1908	39.23	78
3	28	Expanded analyzing portal	Lisa Coffey	Fiction	1941	37.51	79
4	29	Quality-focused multi-tasking challenge	Katrina Underwood	Fiction	1905	31.12	100
5	31	Implemented encompassing conglomeration	Melissa Taylor	Fiction	2010	21.23	44
6	39	Optimized national process improvement	Megan Goodwin	Fiction	1978	10.99	42
7	40	Adaptive didactic interface	Natalie Gonzalez	Fiction	1923	25.97	94
8	47	Reverse-engineered directional conglomeration	John Christian	Fiction	2006	20.37	90
9	62	Re-contextualized real-time strategy	Nicole Lynch	Fiction	1953	26.34	23
10	63	Polarized heuristic database	Franklin Mack	Fiction	1989	22.38	56

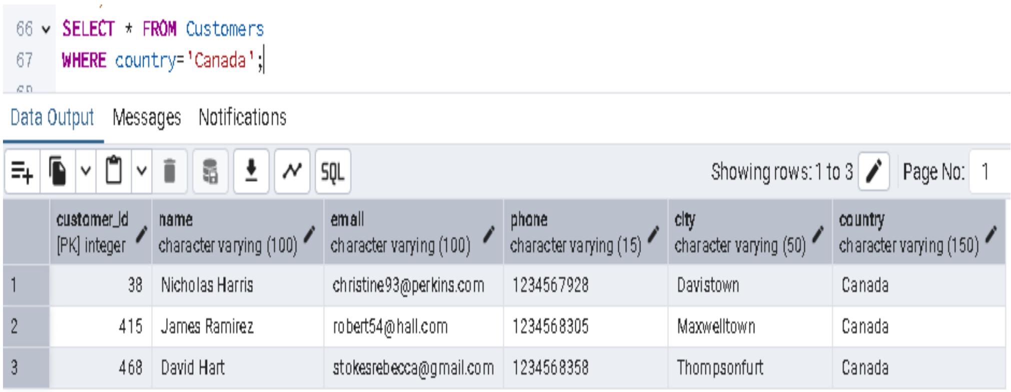
-- 2) Retrieve the total stock of books available:



```
SELECT SUM(stock) AS Total_Stock
From Books;
```

	total_stock bigint
1	25056

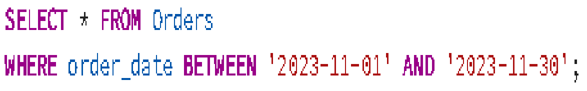
-- 3) List all customers from the Canada:



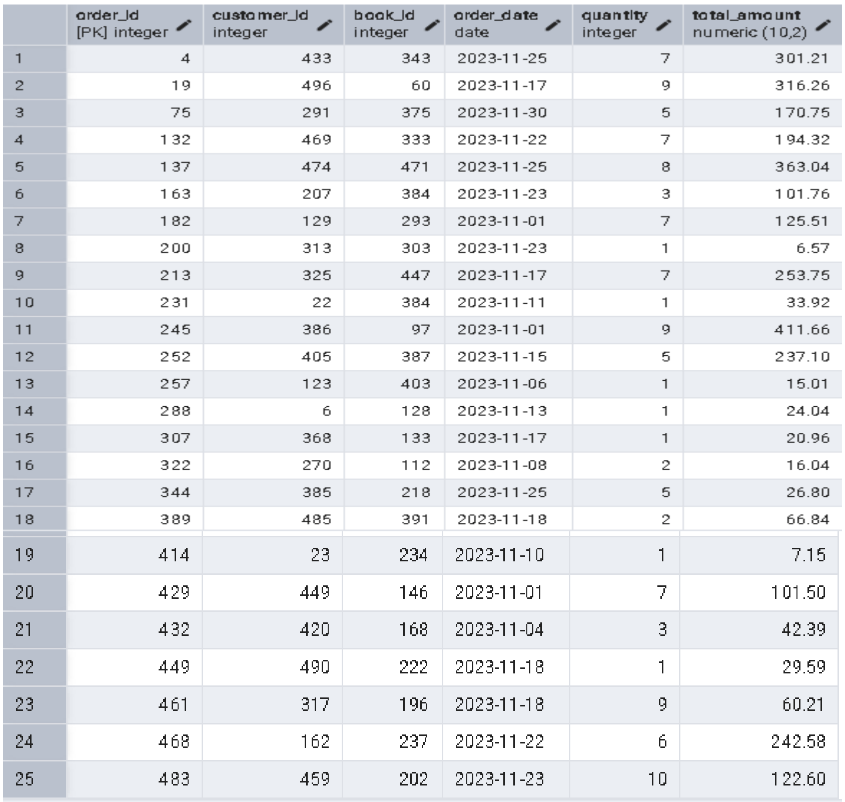
```
SELECT * FROM Customers
WHERE country='Canada';
```

	customer_id [PK] integer	name character varying (100)	email character varying (100)	phone character varying (15)	city character varying (50)	country character varying (150)
1	38	Nicholas Harris	christine93@perkins.com	1234567928	Davistown	Canada
2	415	James Ramirez	robert54@hall.com	1234568305	Maxwelltown	Canada
3	468	David Hart	stokesrebecca@gmail.com	1234568358	Thompsonfurt	Canada

-- 4) Show orders placed in November 2023:

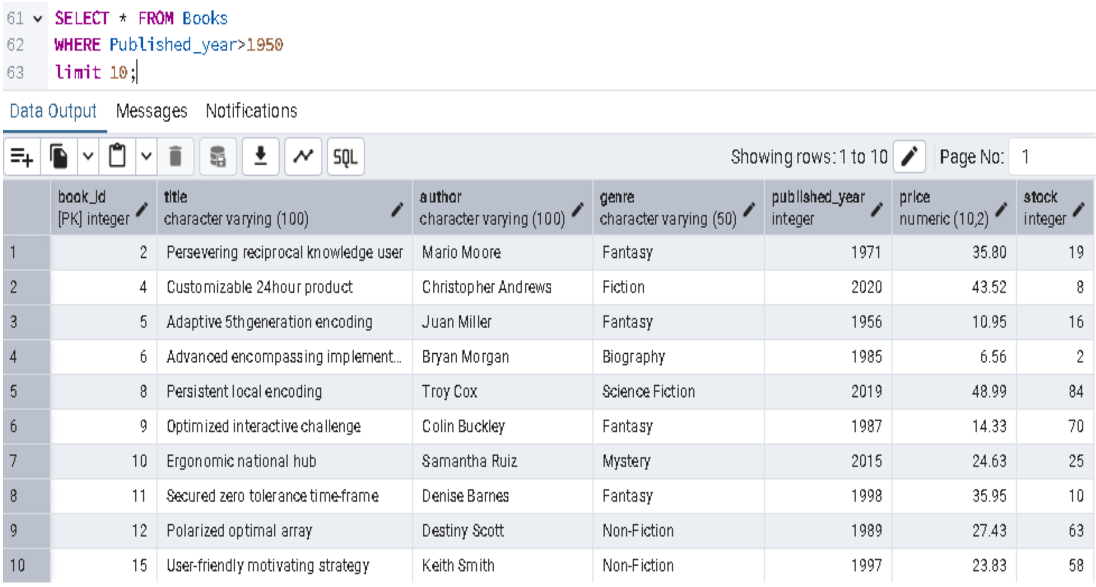


```
SELECT * FROM Orders
WHERE order_date BETWEEN '2023-11-01' AND '2023-11-30';
```



	order_id [PK] integer	customer_id integer	book_id integer	order_date date	quantity integer	total_amount numeric (10,2)
1	4	433	343	2023-11-25	7	301.21
2	19	496	60	2023-11-17	9	316.26
3	75	291	375	2023-11-30	5	170.75
4	132	469	333	2023-11-22	7	194.32
5	137	474	471	2023-11-25	8	363.04
6	163	207	384	2023-11-23	3	101.76
7	182	129	293	2023-11-01	7	125.51
8	200	313	303	2023-11-23	1	6.57
9	213	325	447	2023-11-17	7	253.75
10	231	22	384	2023-11-11	1	33.92
11	245	386	97	2023-11-01	9	411.66
12	252	405	387	2023-11-15	5	237.10
13	257	123	403	2023-11-06	1	15.01
14	288	6	128	2023-11-13	1	24.04
15	307	368	133	2023-11-17	1	20.96
16	322	270	112	2023-11-08	2	16.04
17	344	385	218	2023-11-25	5	26.80
18	389	485	391	2023-11-18	2	66.84
19	414	23	234	2023-11-10	1	7.15
20	429	449	146	2023-11-01	7	101.50
21	432	420	168	2023-11-04	3	42.39
22	449	490	222	2023-11-18	1	29.59
23	461	317	196	2023-11-18	9	60.21
24	468	162	237	2023-11-22	6	242.58
25	483	459	202	2023-11-23	10	122.60

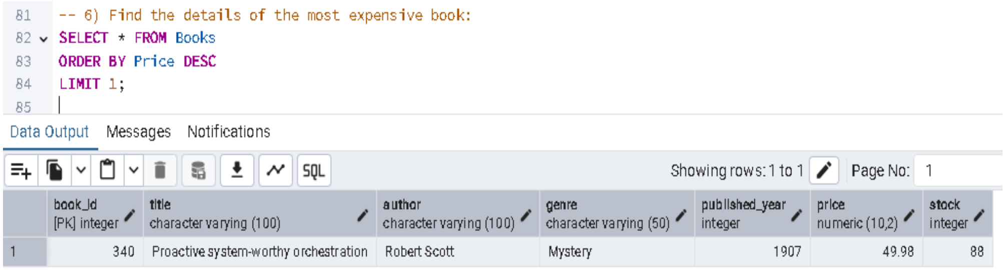
-- 5) Find 10 books published after the year 1950:



```
SELECT * FROM Books
WHERE Published_year>1950
limit 10;
```

	book_id [PK] integer	title character varying (100)	author character varying (100)	genre character varying (50)	published_year integer	price numeric (10,2)	stock integer
1	2	Persevering reciprocal knowledge user	Mario Moore	Fantasy	1971	35.80	19
2	4	Customizable 24hour product	Christopher Andrews	Fiction	2020	43.52	8
3	5	Adaptive 5thgeneration encoding	Juan Miller	Fantasy	1956	10.95	16
4	6	Advanced encompassing implement...	Bryan Morgan	Biography	1985	6.56	2
5	8	Persistent local encoding	Troy Cox	Science Fiction	2019	48.99	84
6	9	Optimized interactive challenge	Colin Buckley	Fantasy	1987	14.33	70
7	10	Ergonomic national hub	Samantha Ruiz	Mystery	2015	24.63	25
8	11	Secured zero tolerance time-frame	Denise Barnes	Fantasy	1998	35.95	10
9	12	Polarized optimal array	Destiny Scott	Non-Fiction	1989	27.43	63
10	15	User-friendly motivating strategy	Keith Smith	Non-Fiction	1997	23.83	58

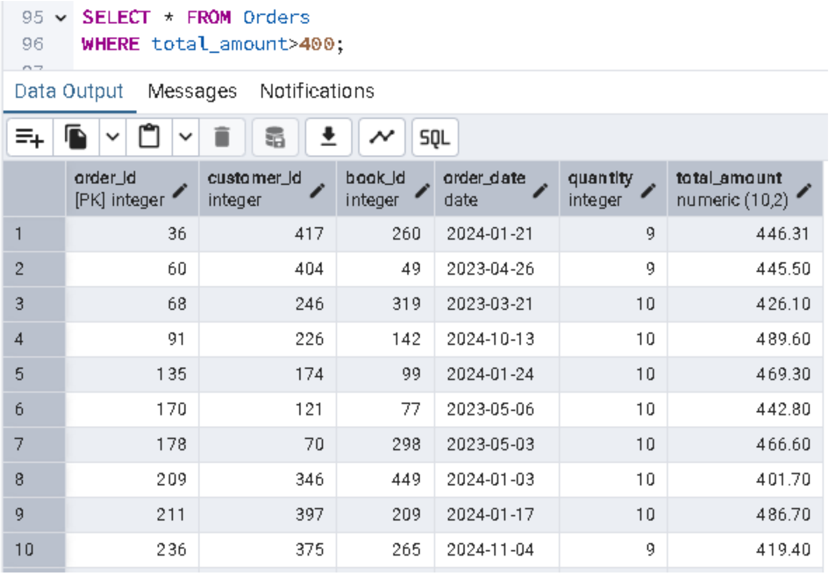
-- 6) Find the details of the most expensive book:



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

	book_id [PK] integer	title character varying (100)	author character varying (100)	genre character varying (50)	published_year integer	price numeric (10,2)	stock integer
1	340	Proactive system-worthy orchestration	Robert Scott	Mystery	1907	49.98	88

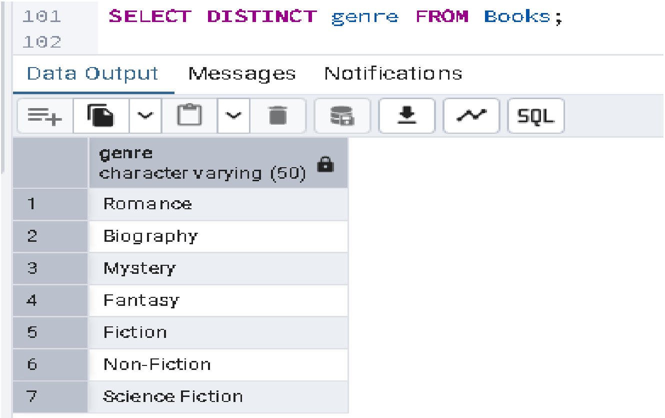
-- 7) Retrieve all orders where the total amount exceeds $400:



```sql
SELECT * FROM Orders
WHERE total_amount>400;
```

	order_id [PK] integer	customer_id integer	book_id integer	order_date date	quantity integer	total_amount numeric (10,2)
1	36	417	260	2024-01-21	9	446.31
2	60	404	49	2023-04-26	9	445.50
3	68	246	319	2023-03-21	10	426.10
4	91	226	142	2024-10-13	10	489.60
5	135	174	99	2024-01-24	10	469.30
6	170	121	77	2023-05-06	10	442.80
7	178	70	298	2023-05-03	10	466.60
8	209	346	449	2024-01-03	10	401.70
9	211	397	209	2024-01-17	10	486.70
10	236	375	265	2024-11-04	9	419.40

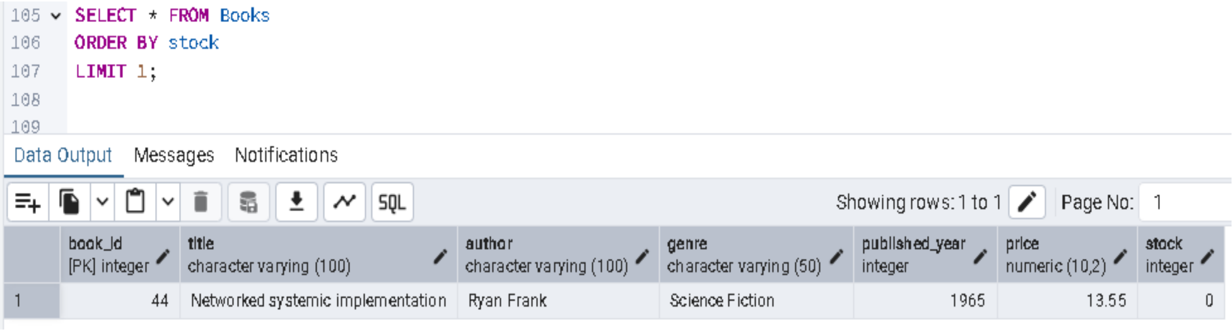
-- 8) List all genres available in the Books table:



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

	genre character varying (50)
1	Romance
2	Biography
3	Mystery
4	Fantasy
5	Fiction
6	Non-Fiction
7	Science Fiction

-- 9) Find the book with the lowest stock:

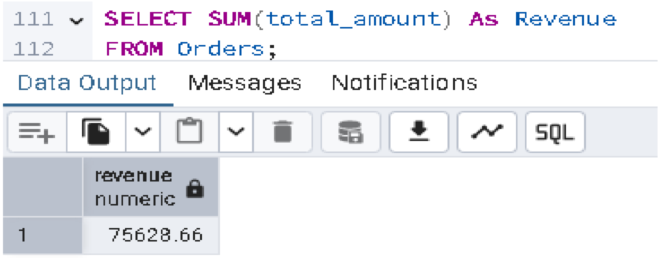


```sql
SELECT * FROM Books
ORDER BY stock
LIMIT 1;
```

Showing rows: 1 to 1 Page No: 1

	book_id [PK] integer	title character varying (100)	author character varying (100)	genre character varying (50)	published_year integer	price numeric (10,2)	stock integer
1	44	Networked systemic implementation	Ryan Frank	Science Fiction	1965	13.55	0

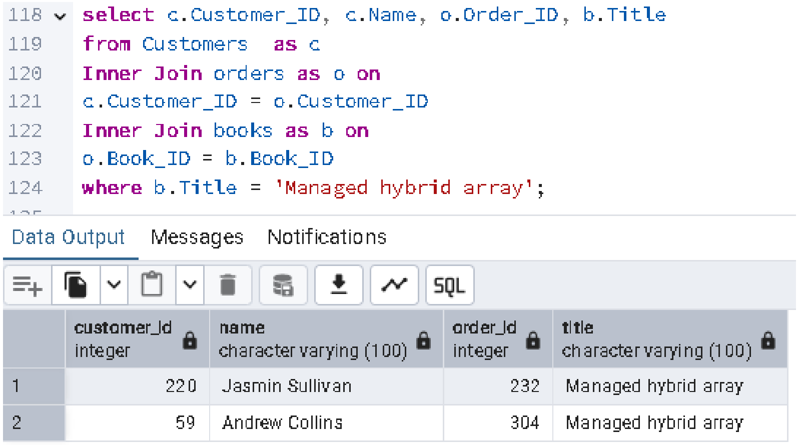
-- 10) Calculate the total revenue generated from all orders:



```sql
SELECT SUM(total_amount) As Revenue
FROM Orders;
```

	revenue numeric
1	75628.66

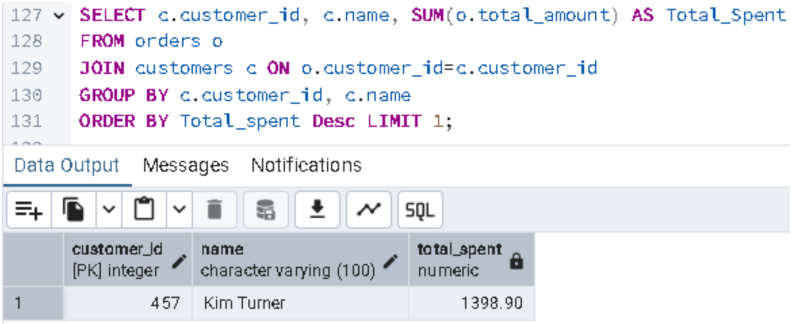
--11) Retrieve customer names who have ordered the book titled "Managed hybrid array":



```sql
select c.Customer_ID, c.Name, o.Order_ID, b.Title
from Customers  as c
Inner Join orders as o on
c.Customer_ID = o.Customer_ID
Inner Join books as b on
o.Book_ID = b.Book_ID
where b.Title = 'Managed hybrid array';
```

	customer_id integer	name character varying (100)	order_id integer	title character varying (100)
1	220	Jasmin Sullivan	232	Managed hybrid array
2	59	Andrew Collins	304	Managed hybrid array

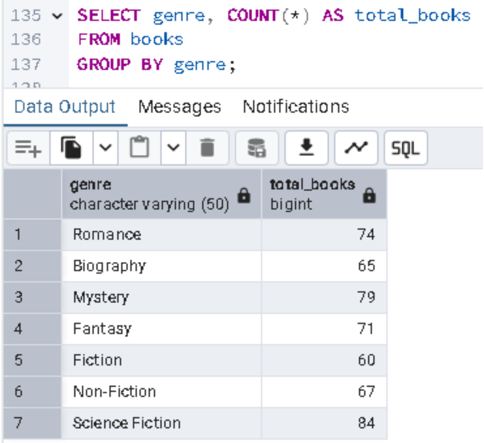
--12) Find the customer who spent the most on orders:



```sql
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 1;
```

	customer_id [PK] integer	name character varying (100)	total_spent numeric
1	457	Kim Turner	1398.90

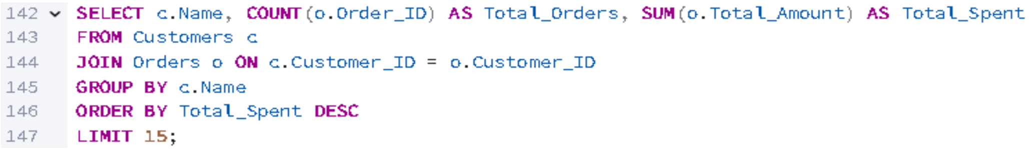
--13) Find the total number of books available in each genre:



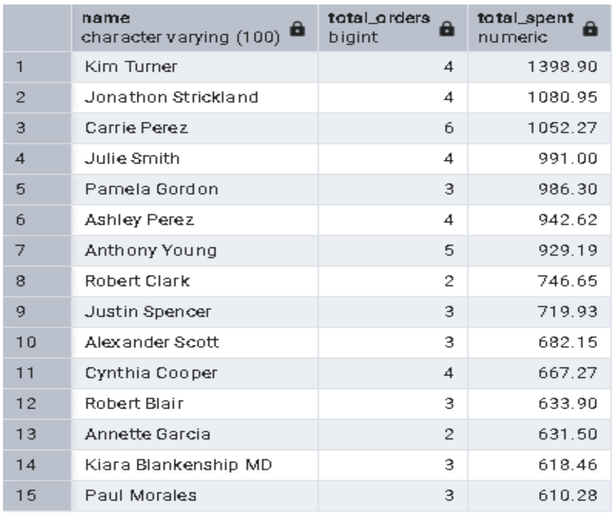
```sql
SELECT genre, COUNT(*) AS total_books
FROM books
GROUP BY genre;
```

	genre character varying (50)	total_books bigint
1	Romance	74
2	Biography	65
3	Mystery	79
4	Fantasy	71
5	Fiction	60
6	Non-Fiction	67
7	Science Fiction	84

--14) Top 15 Customers based on total amount and Spent along with their total orders count:

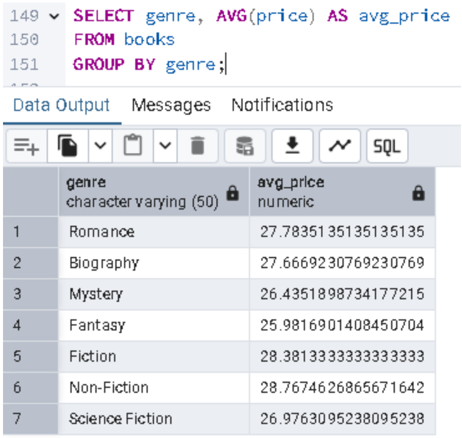


```
SELECT c.Name, COUNT(o.Order_ID) AS Total_Orders, SUM(o.Total_Amount) AS Total_Spent
FROM Customers c
JOIN Orders o ON c.Customer_ID = o.Customer_ID
GROUP BY c.Name
ORDER BY Total_Spent DESC
LIMIT 15;
```



	name character varying (100)	total_orders bigint	total_spent numeric
1	Kim Turner	4	1398.90
2	Jonathon Strickland	4	1080.95
3	Carrie Perez	6	1052.27
4	Julie Smith	4	991.00
5	Pamela Gordon	3	986.30
6	Ashley Perez	4	942.62
7	Anthony Young	5	929.19
8	Robert Clark	2	746.65
9	Justin Spencer	3	719.93
10	Alexander Scott	3	682.15
11	Cynthia Cooper	4	667.27
12	Robert Blair	3	633.90
13	Annette Garcia	2	631.50
14	Kiara Blankenship MD	3	618.46
15	Paul Morales	3	610.28

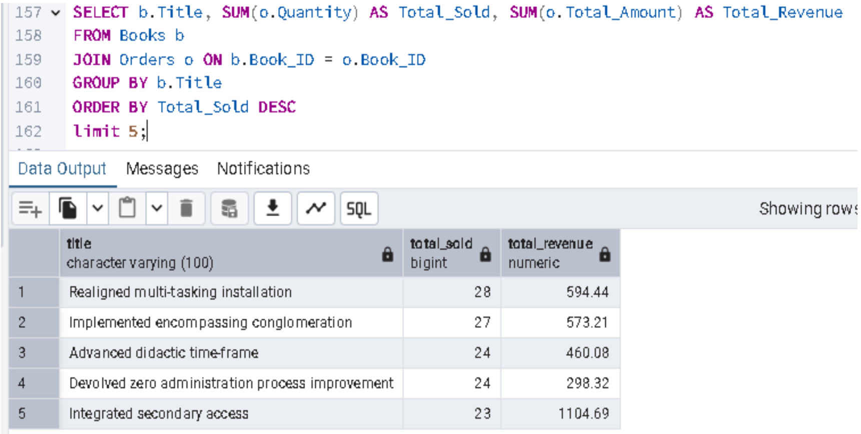
--15) Find the average price of books per genre:



```
SELECT genre, AVG(price) AS avg_price
FROM books
GROUP BY genre;
```

	genre character varying (50)	avg_price numeric
1	Romance	27.7835135135135135
2	Biography	27.6669230769230769
3	Mystery	26.4351898734177215
4	Fantasy	25.9816901408450704
5	Fiction	28.3813333333333333
6	Non-Fiction	28.7674626865671642
7	Science Fiction	26.9763095238095238

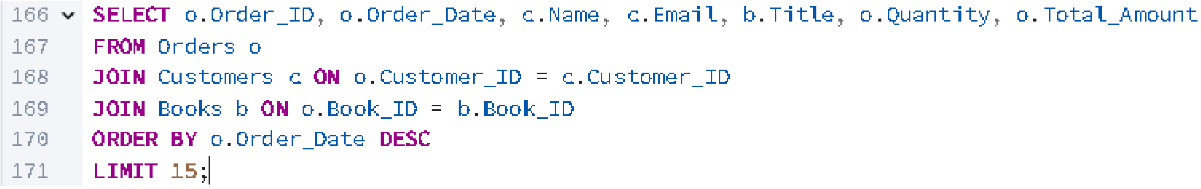
--16) Top 5 Selling Books with Total Sales



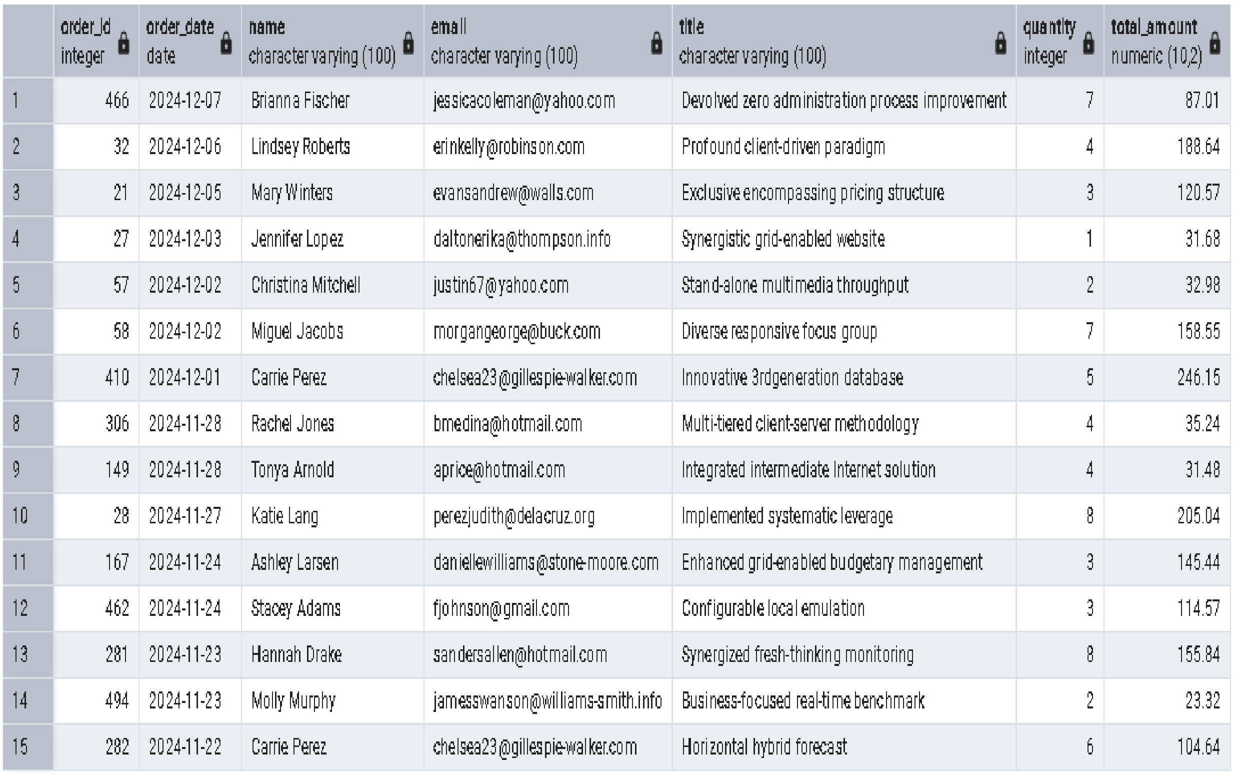
```
SELECT b.Title, SUM(o.Quantity) AS Total_Sold, SUM(o.Total_Amount) AS Total_Revenue
FROM Books b
JOIN Orders o ON b.Book_ID = o.Book_ID
GROUP BY b.Title
ORDER BY Total_Sold DESC
limit 5;
```

	title character varying (100)	total_sold bigint	total_revenue numeric
1	Realigned multi-tasking installation	28	594.44
2	Implemented encompassing conglomeration	27	573.21
3	Advanced didactic time-frame	24	460.08
4	Devolved zero administration process improvement	24	298.32
5	Integrated secondary access	23	1104.69

--17) Orders with Customer Details (Recent 15 Orders)

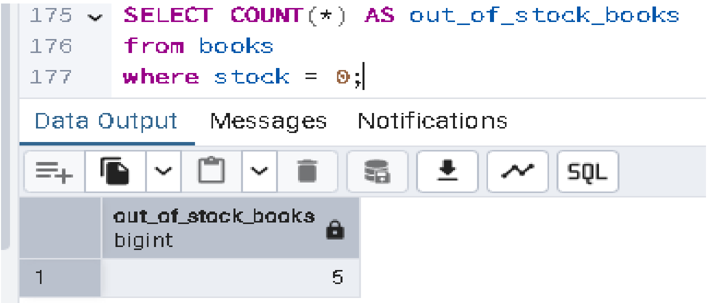


```sql
SELECT o.Order_ID, o.Order_Date, c.Name, c.Email, b.Title, o.Quantity, o.Total_Amount
FROM Orders o
JOIN Customers c ON o.Customer_ID = c.Customer_ID
JOIN Books b ON o.Book_ID = b.Book_ID
ORDER BY o.Order_Date DESC
LIMIT 15;
```



	order_id integer	order_date date	name character varying (100)	email character varying (100)	title character varying (100)	quantity integer	total_amount numeric (10,2)
1	466	2024-12-07	Brianna Fischer	jessicacoleman@yahoo.com	Devolved zero administration process improvement	7	87.01
2	32	2024-12-06	Lindsey Roberts	erinkelly@robinson.com	Profound client-driven paradigm	4	188.64
3	21	2024-12-05	Mary Winters	evansandrew@walls.com	Exclusive encompassing pricing structure	3	120.57
4	27	2024-12-03	Jennifer Lopez	daltonerika@thompson.info	Synergistic grid-enabled website	1	31.68
5	57	2024-12-02	Christina Mitchell	justin67@yahoo.com	Stand-alone multimedia throughput	2	32.98
6	58	2024-12-02	Miguel Jacobs	morgangeorge@buck.com	Diverse responsive focus group	7	158.55
7	410	2024-12-01	Carrie Perez	chelsea23@gillespie-walker.com	Innovative 3rdgeneration database	5	246.15
8	306	2024-11-28	Rachel Jones	bmedina@hotmail.com	Multi-tiered client-server methodology	4	35.24
9	149	2024-11-28	Tonya Arnold	aprice@hotmail.com	Integrated intermediate Internet solution	4	31.48
10	28	2024-11-27	Katie Lang	perezjudith@delacruz.org	Implemented systematic leverage	8	205.04
11	167	2024-11-24	Ashley Larsen	daniellewilliams@stone-moore.com	Enhanced grid-enabled budgetary management	3	145.44
12	462	2024-11-24	Stacey Adams	fjohnson@gmail.com	Configurable local emulation	3	114.57
13	281	2024-11-23	Hannah Drake	sandersallen@hotmail.com	Synergized fresh-thinking monitoring	8	155.84
14	494	2024-11-23	Molly Murphy	jamesswanson@williams-smith.info	Business-focused real-time benchmark	2	23.32
15	282	2024-11-22	Carrie Perez	chelsea23@gillespie-walker.com	Horizontal hybrid forecast	6	104.64

--18) Find the total number of books that are out of stock:



```sql
SELECT COUNT(*) AS out_of_stock_books
from books
where stock = 0;
```

	out_of_stock_books bigint
1	5

Conclusion

The **Online Book Store Database** successfully provides a structured way to manage books, customers, orders, and payments. By using SQL queries, we:

- ✓ **Optimized inventory management** through stock tracking and book ranking.
- ✓ **Analyzed customer behavior** to identify loyal customers and recent purchases.
- ✓ **Generated sales insights** to support data-driven business decision-making.
- ✓ **Improved database performance** using indexing and query optimization techniques.

This project demonstrates how **SQL is crucial** in handling and analyzing large datasets efficiently in an e-commerce environment. The same approach can be applied to **real-world e-commerce applications.**