Online Book Store

SQL Project For Data Analysis

INTRODUCTION

1. This SQL project is based on a fictional online bookstore. It uses real-time data queries to analyse customer behaviour, book performance, and sales trends. The purpose is to gain actionable insights and strengthen SQL skills through hands-on analysis.
2. This project also focuses on enhancing data analysis skills by writing optimized SQL queries that solve real-world business questions and support decision-making processes.

OBJECTIVE

To demonstrate practical SQL skills through a real-world dataset, especially in the absence of formal job experience.

To analyse and manage data from an Online Bookstore system using structured queries

1. To showcase practical SQL skills by working with a real-world bookstore dataset.

2. To perform data analysis using SELECT, JOIN, GROUP BY, and aggregate functions.

3. To extract business insights such as top-selling books, revenue, and customer behaviour.

4. To simulate real-time reporting and decision-making scenarios using SQL queries.

5. To build a portfolio project that demonstrates SQL proficiency to potential Skills.

ANALYSIS

QUESTION 1. Retrieve all books in the "Fiction" genre?

SELECT * FROM Books WHERE Genre='Fiction';

This filters the Books table and returns all rows where the genre is “Fiction”. Useful for analysing only fiction books’ details like price, stock, author, etc.

Data Output Messages Notifications

Showing rows: 1 t

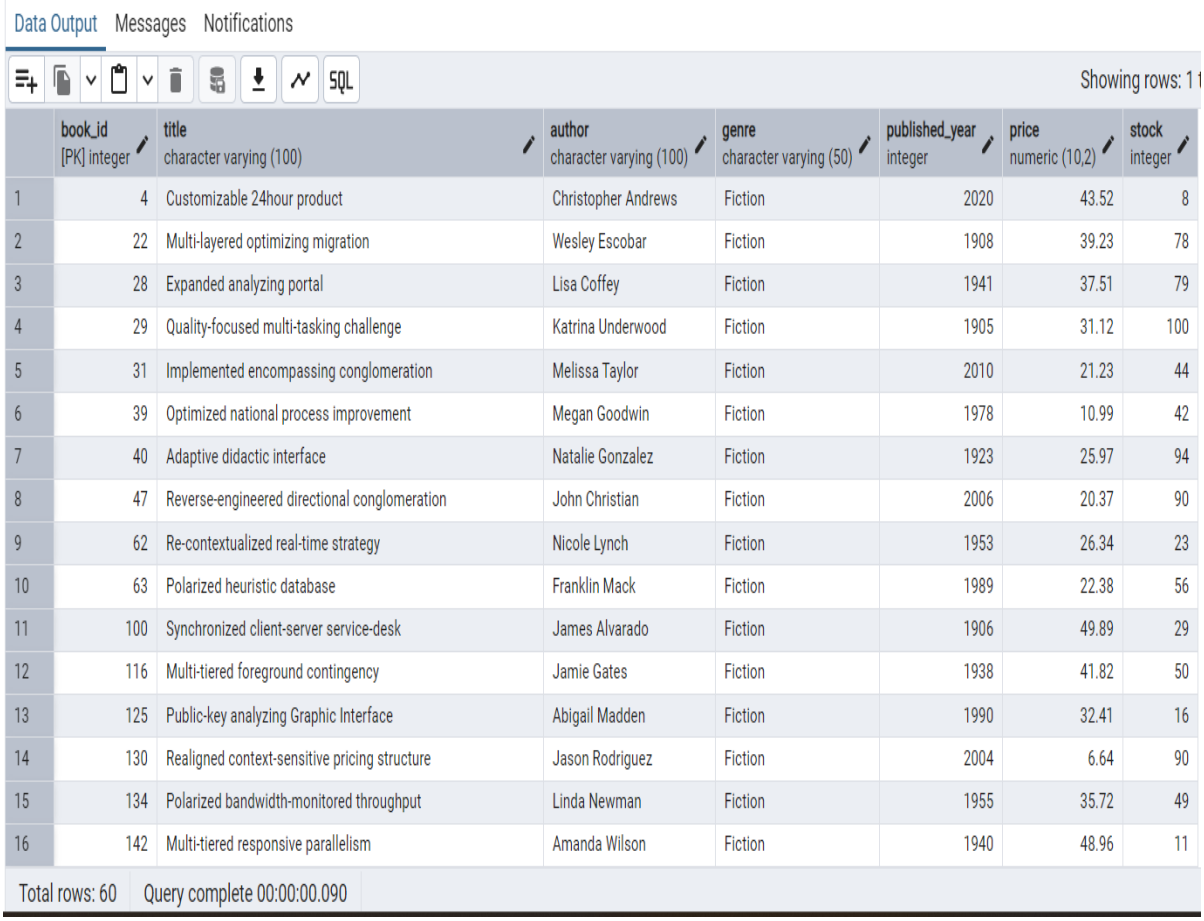
	book_id [PK] integer	title character varying (100)	author character varying (100)	genre character varying (50)	published_year integer	price numeric (10,2)	stock integer
1	4	Customizable 24hour product	Christopher Andrews	Fiction	2020	43.52	8
2	22	Multi-layered optimizing migration	Wesley Escobar	Fiction	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
11	100	Synchronized client-server service-desk	James Alvarado	Fiction	1906	49.89	29
12	116	Multi-tiered foreground contingency	Jamie Gates	Fiction	1938	41.82	50
13	125	Public-key analyzing Graphic Interface	Abigail Madden	Fiction	1990	32.41	16
14	130	Realigned context-sensitive pricing structure	Jason Rodriguez	Fiction	2004	6.64	90
15	134	Polarized bandwidth-monitored throughput	Linda Newman	Fiction	1955	35.72	49
16	142	Multi-tiered responsive parallelism	Amanda Wilson	Fiction	1940	48.96	11

Total rows: 60 Query complete 00:00:00.090

2. Find books published after the year 1950?

SELECT * FROM Books WHERE Published year > 1950;

Description: Returns books where the Published year is greater than 1950. Helps to study modern books or analyse recent trends.



Data Output Messages Notifications

Showing rows: 1 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	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
11	100	Synchronized client-server service-desk	James Alvarado	Fiction	1906	49.89	29
12	116	Multi-tiered foreground contingency	Jamie Gates	Fiction	1938	41.82	50
13	125	Public-key analyzing Graphic Interface	Abigail Madden	Fiction	1990	32.41	16
14	130	Realigned context-sensitive pricing structure	Jason Rodriguez	Fiction	2004	6.64	90
15	134	Polarized bandwidth-monitored throughput	Linda Newman	Fiction	1955	35.72	49
16	142	Multi-tiered responsive parallelism	Amanda Wilson	Fiction	1940	48.96	11

Total rows: 60 Query complete 00:00:00.090

QUESTION 3. List all customers from Canada?

SELECT * FROM Customers WHERE country='Canada';

Description: Filters and shows only those customers who are from **Canada**. Great for country-wise customer analysis.

Data Output Messages Notifications

	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?

Description: Returns all orders made during **November 2023**. Helps in month-wise sales performance analysis.

Data Output Messages Notifications

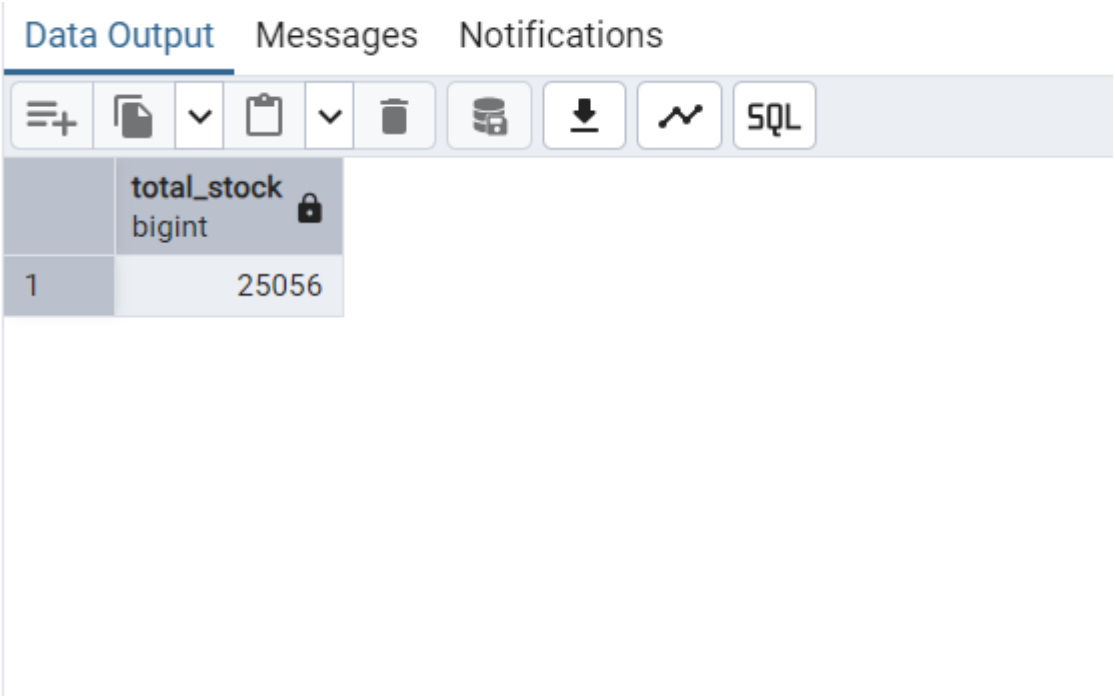
	order_id [PK] integer	customer_id integer	book_id integer	order_date date	quantity integer	total_amount numeric (10,2)
1	4	433	343	2023-11-25	7	301.21
2	19	496	60	2023-11-17	9	316.26
3	75	291	375	2023-11-30	5	170.75
4	132	469	333	2023-11-22	7	194.32
5	137	474	471	2023-11-25	8	363.04
6	163	207	384	2023-11-23	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

Total rows: 25 Query complete 00:00:00.099

5. Question: Retrieve the total stock of books available.

SELECT SUM(stock) AS Total_Stock FROM Books;

Description: Calculates the total number of books in stock by summing the stock column.



6. Question: Find the details of the most expensive book.

SELECT * FROM Books ORDER BY Price DESC LIMIT 10;

Description: Fetches the top 10 most expensive books from the Books table.

	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
2	155	Optimized content-based standardization	Timothy Adams	Science Fiction	1901	49.96	88
3	240	Stand-alone content-based hub	Lisa Ellis	Fantasy	1957	49.90	41
4	100	Synchronized client-server service-desk	James Alvarado	Fiction	1906	49.89	29
5	119	Switchable modular moratorium	Tonya Saunders	Romance	2010	49.88	76
6	260	Business-focused methodical application	Brian King	Fiction	1907	49.59	10
7	95	Function-based heuristic analyzer	Craig Smith	Biography	1990	49.53	11
8	49	Robust attitude-oriented attitude	Zachary Hayes	Biography	1955	49.50	15
9	141	Exclusive hybrid monitoring	Dr. Heather Boyd	Romance	1909	49.42	36
10	360	Profound tertiary encoding	Jennifer Jones	Non-Fiction	2020	49.34	73

7. Question: Show all customers who ordered more than 1 quantity of a book.

SELECT * FROM Orders WHERE quantity > 1;

Description: Displays all orders where the quantity ordered is greater than one.

Data Output | Messages | Notifications

	order_id [PK] integer	customer_id integer	book_id integer	order_date date	quantity integer	total_amount numeric (10,2)
1	1	84	169	2023-05-26	8	188.56
2	2	137	301	2023-01-23	10	216.60
3	3	216	261	2024-05-27	6	85.50
4	4	433	343	2023-11-25	7	301.21
5	5	14	431	2023-07-26	7	136.36
6	6	439	119	2024-10-11	5	249.40
7	7	195	467	2023-10-23	6	82.92
8	8	32	159	2024-05-07	4	144.84
9	9	109	407	2024-01-04	9	379.71
10	10	94	122	2024-07-09	4	123.00
11	12	454	3	2024-06-17	2	31.50
12	13	420	180	2023-06-08	5	125.45
13	14	454	319	2023-08-24	2	85.22
14	15	127	479	2023-01-10	6	229.62
15	16	412	196	2023-10-06	8	53.52
16	17	462	481	2023-03-20	5	52.75

Total rows: 438 | Query complete 00:00:00.194

8. Question: Retrieve all orders where the total amount exceeds $20.

SELECT * FROM Orders WHERE total_amount > 20;

Description: Shows orders where the total_amount is more than $20.

Data Output | Messages | Notifications

	order_id [PK] integer	customer_id integer	book_id integer	order_date date	quantity integer	total_amount numeric (10,2)
1	1	84	169	2023-05-26	8	188.56
2	2	137	301	2023-01-23	10	216.60
3	3	216	261	2024-05-27	6	85.50
4	4	433	343	2023-11-25	7	301.21
5	5	14	431	2023-07-26	7	136.36
6	6	439	119	2024-10-11	5	249.40
7	7	195	467	2023-10-23	6	82.92
8	8	32	159	2024-05-07	4	144.84
9	9	109	407	2024-01-04	9	379.71
10	10	94	122	2024-07-09	4	123.00
11	11	131	206	2023-10-16	1	38.01
12	12	454	3	2024-06-17	2	31.50
13	13	420	180	2023-06-08	5	125.45
14	14	454	319	2023-08-24	2	85.22
15	15	127	479	2023-01-10	6	229.62
16	16	412	196	2023-10-06	8	53.52

Total rows: 473 | Query complete 00:00:00.134

9. Question: List all genres available in the Books table.

SELECT DISTINCT genre FROM Books;

Description: Returns a list of all unique genres found in the Books table.

Data Output Messages Notifications

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

10. Question: Find the book with the lowest stock.

SELECT * FROM Books ORDER BY stock LIMIT 10;

Description: Retrieves the book that has the least quantity in stock.

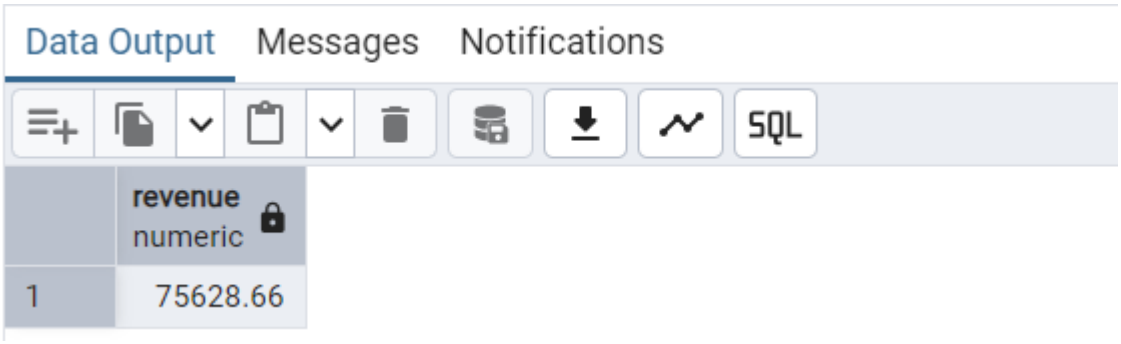
Data Output Messages Notifications

	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	127	Business-focused real-time benchmark	David Nelson	Science Fiction	1997	11.66	0
2	44	Networked systemic implementation	Ryan Frank	Science Fiction	1965	13.55	0
3	378	Future-proofed heuristic function	Samantha Mcclain	Romance	1903	6.01	0
4	60	Robust eco-centric capacity	Brian Haney	Biography	1990	35.14	0
5	163	Object-based eco-centric challenge	Douglas Mccarthy	Non-Fiction	1905	19.11	0
6	137	Networked contextually-based encrypti...	Leslie Montoya	Biography	1942	26.33	1
7	259	Pre-emptive incremental secured line	Jason Perez	Science Fiction	1994	14.48	1
8	19	Progressive asymmetric Internet solution	Sean Miller	Science Fiction	1990	11.31	1
9	199	Configurable fault-tolerant interface	Mark Parker	Mystery	1923	42.81	1
10	232	Monitored 24/7 groupware	Vicki Mcgee	Science Fiction	1925	9.62	1

11. Question: Calculate the total revenue generated from all orders.

SELECT SUM(total_amount) As Revenue FROM Orders;

Description: Calculates the total revenue by summing up the total amount from all orders



	revenue numeric
1	75628.66

12 Question . Retrieve the total number of books sold for each genre?

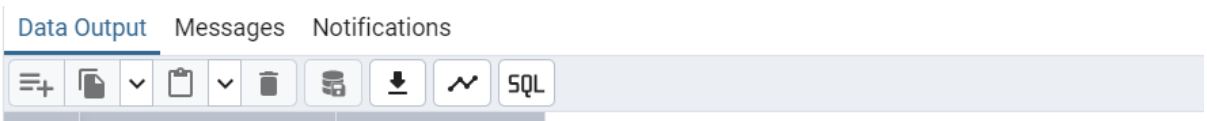
SELECT b.genre, SUM(o.quantity) AS total_books_sold

FROM orders o

JOIN books b ON o.book_id = b.book_id

GROUP BY b.genre;

Description: Joins Orders and Books tables to find how many books were sold in each genre.

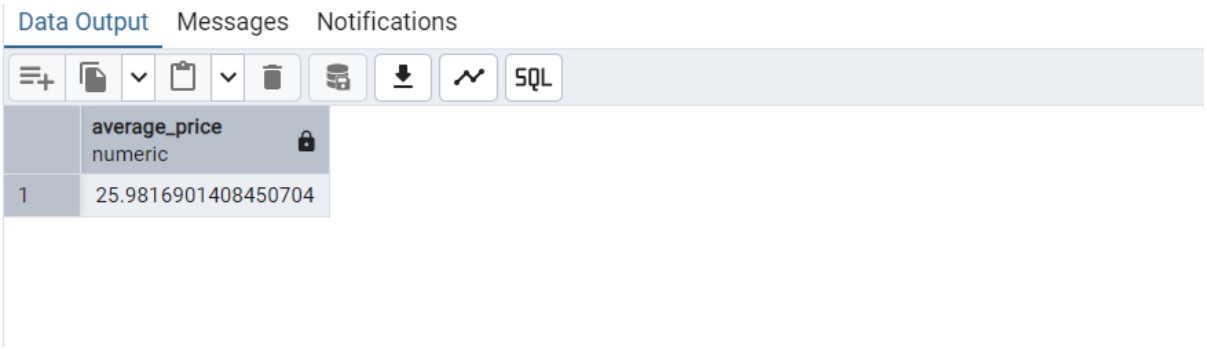


	genre character varying (50)	total_books_sold bigint
1	Romance	439
2	Biography	285
3	Mystery	504
4	Fantasy	446
5	Fiction	225
6	Non-Fiction	351
7	Science Fiction	447

13. Find the average price of books in the "Fantasy" genre?

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

Description: Calculates the average price of books categorized under the 'Fantasy' genre.


Data Output Messages Notifications

SQL

	average_price numeric
1	25.9816901408450704

14. List customers who have placed at least 2 orders

SELECT o.customer_id, c.name, COUNT(o.Order_id) AS ORDER_COUNT

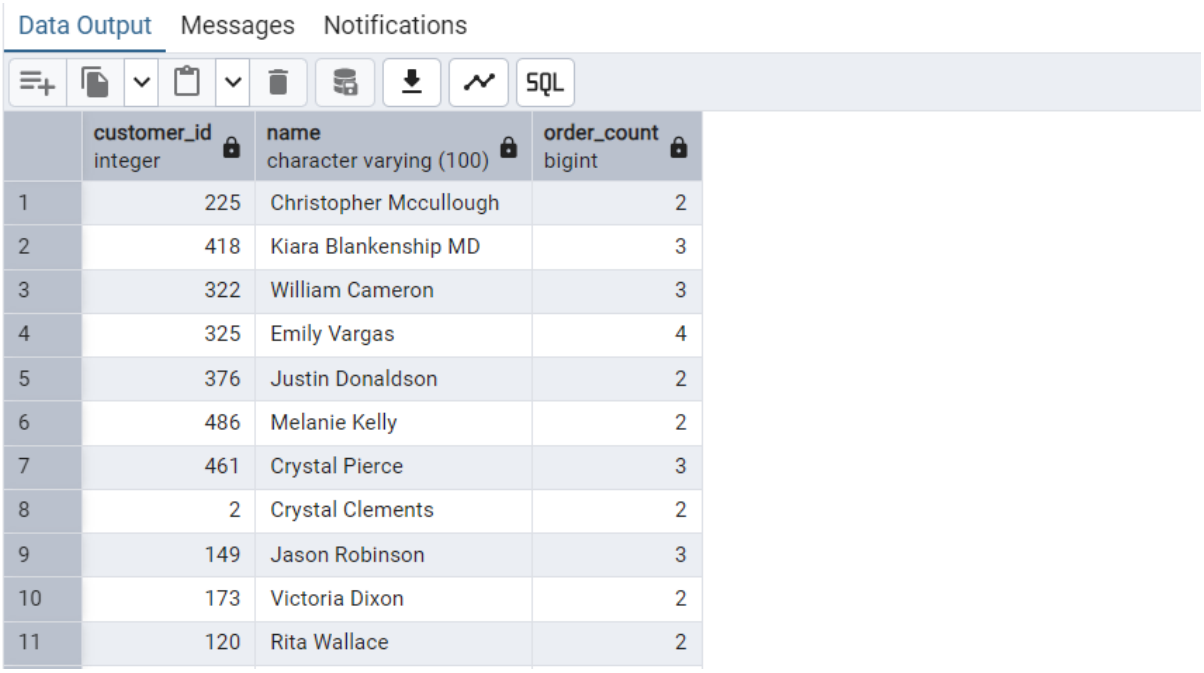
FROM orders o

JOIN customers c ON o.customer_id = c.customer_id

GROUP BY o.customer_id, c.name

HAVING COUNT(Order_id) >= 2;

Description: Identifies customers who placed two or more orders.


Data Output Messages Notifications

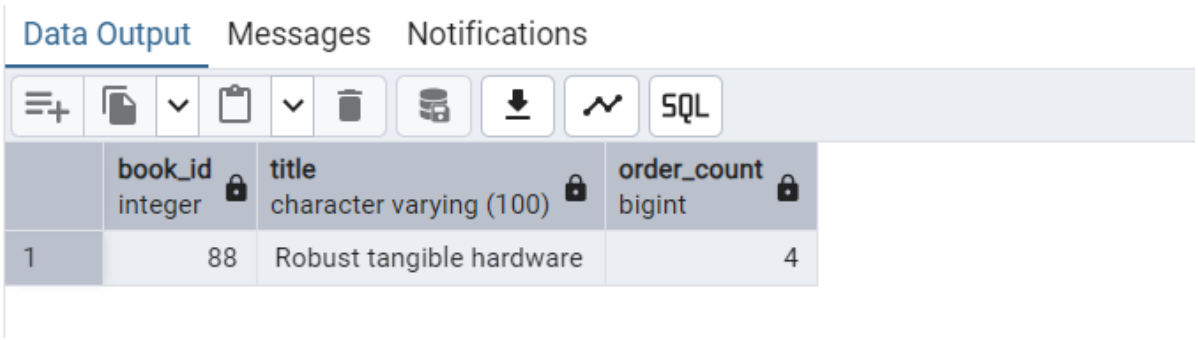
SQL

	customer_id integer	name character varying (100)	order_count bigint
1	225	Christopher Mccullough	2
2	418	Kiara Blankenship MD	3
3	322	William Cameron	3
4	325	Emily Vargas	4
5	376	Justin Donaldson	2
6	486	Melanie Kelly	2
7	461	Crystal Pierce	3
8	2	Crystal Clements	2
9	149	Jason Robinson	3
10	173	Victoria Dixon	2
11	120	Rita Wallace	2

15. Find the most frequently ordered book?

```
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 1;
```

Description: Shows the book with the highest number of orders.



Data Output | Messages | Notifications

	book_id integer	title character varying (100)	order_count bigint
1	88	Robust tangible hardware	4

16. Show the top 10 most expensive books of the 'Fantasy' genre?

```
SELECT * FROM books
WHERE genre = 'Fantasy'
ORDER BY price DESC
LIMIT 10;
```

Description: Lists the 3 costliest books within the 'Fantasy' genre.

Data Output | Messages | Notifications

	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	240	Stand-alone content-based hub	Lisa Ellis	Fantasy	1957	49.90	41
2	462	Innovative 3rdgeneration database	Allison Contreras	Fantasy	1988	49.23	62
3	238	Optimized even-keeled analyzer	Sherri Griffith	Fantasy	1975	48.97	72
4	152	Advanced dynamic complexity	Elizabeth Ayala	Fantasy	1983	46.77	84
5	98	Digitized encompassing definition	Thomas Gonzales	Fantasy	1996	46.71	83
6	72	Networked contextually-based throughput	Andrea Eaton	Fantasy	1959	46.54	65
7	422	Grass-roots systematic moderator	Joyce Patton	Fantasy	1919	45.91	39
8	54	Progressive bi-directional methodology	Adrian Taylor	Fantasy	2015	44.78	27
9	77	Cross-platform asynchronous encryption	Michelle Edwards	Fantasy	1935	44.28	88
10	393	Triple-buffered multimedia access	Nathan Ashley	Fantasy	1987	44.00	44

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;
```

Description: Calculates how many books each author has sold.

Data Output | Messages | Notifications

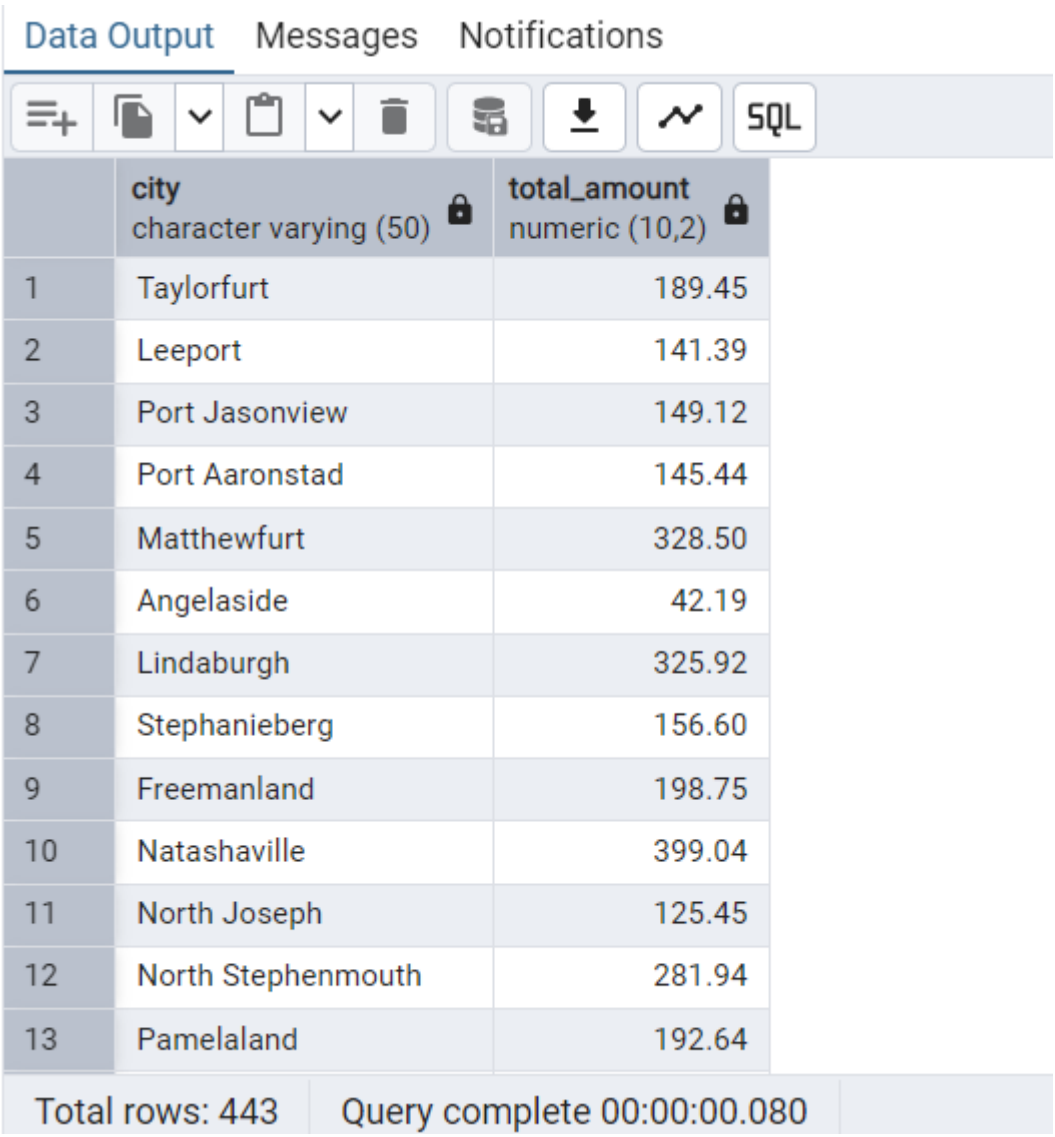
	author character varying (100)	total_books_sold bigint
1	Jared Cortez	10
2	Tracy Parker	11
3	Taylor Wang	9
4	Cathy Knight	6
5	Bianca Matthews	3
6	Douglas Malone	6
7	James Alvarado	9
8	Betty Cross	6
9	Michael Hill	20
10	Steven Mcdonald	15
11	Paul Miles	19
12	Leonard Vega	3
13	Grace Waller	15

Total rows: 314 | Query complete 00:00:00.175

18. 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;
```

Description: Returns cities of customers who placed orders above $30.



Data Output | Messages | Notifications

	city character varying (50)	total_amount numeric (10,2)
1	Taylorfurt	189.45
2	Leeport	141.39
3	Port Jasonview	149.12
4	Port Aaronstad	145.44
5	Matthewfurt	328.50
6	Angelaside	42.19
7	Lindaburgh	325.92
8	Stephanieberg	156.60
9	Freemanland	198.75
10	Natashaville	399.04
11	North Joseph	125.45
12	North Stephenmouth	281.94
13	Pamelaland	192.64

Total rows: 443 Query complete 00:00:00.080

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 10;
```

Description: Finds the top spender among all customers.

Data Output | Messages | Notifications

	customer_id [PK] integer	name character varying (100)	total_spent numeric
1	457	Kim Turner	1398.90
2	174	Jonathon Strickland	1080.95
3	364	Carrie Perez	1052.27
4	405	Julie Smith	991.00
5	386	Pamela Gordon	986.30
6	425	Ashley Perez	942.62
7	474	Anthony Young	929.19
8	163	Robert Clark	746.65
9	167	Justin Spencer	719.93
10	214	Alexander Scott	682.15