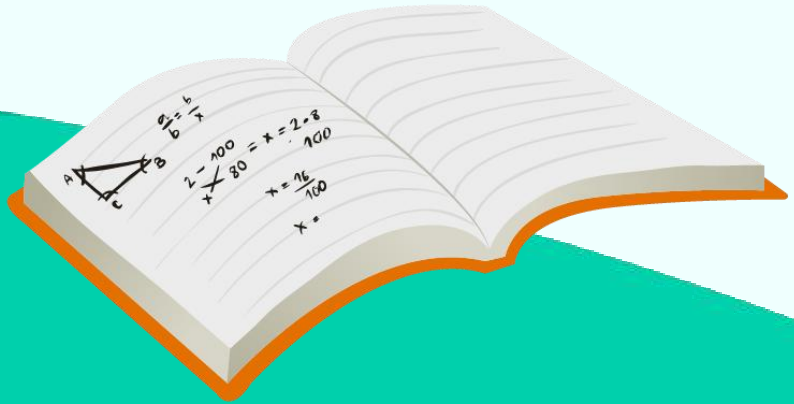
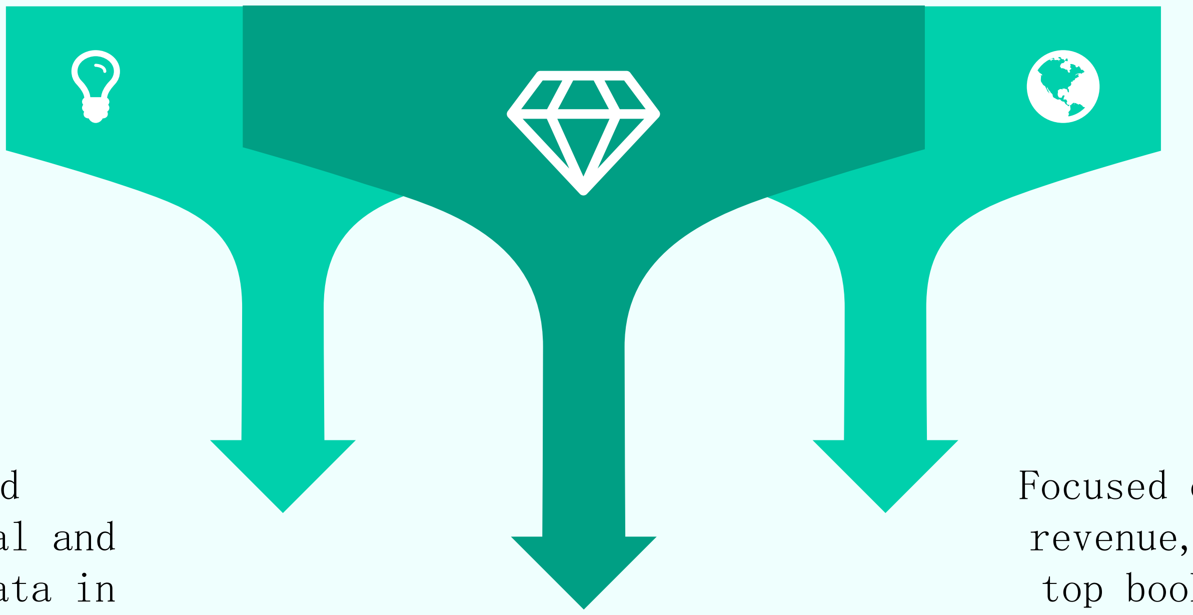
# Online Bookstore SQL Data Analysis

The user can demonstrate on a projector or computer, or print the presentation and make it film  The user can demonstrate on a projector or computer, or print the presentation and make it film



# Project Overview



Analyzed transactional and inventory data in PostgreSQL.

Extracted business insights for e-commerce decision-making.

Focused on KPIs like revenue, stock, and top books/authors.

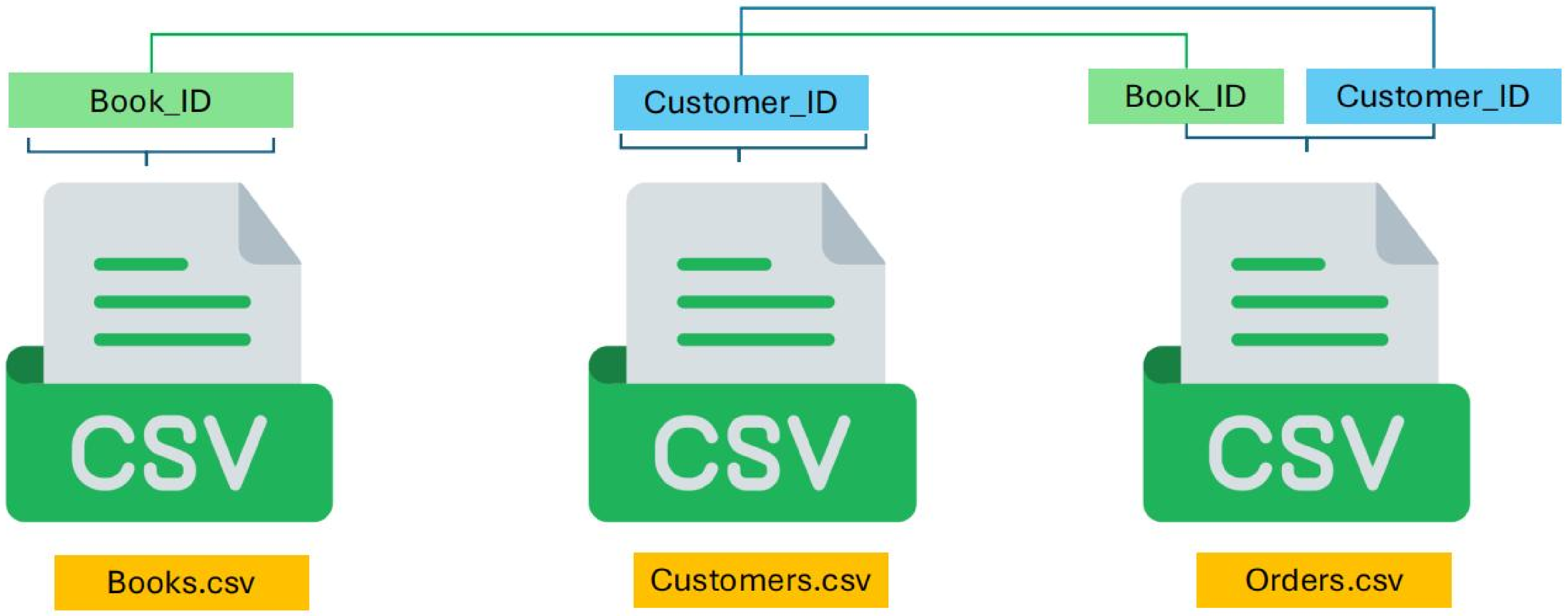
## Datasets Used

1. books.csv – title, author, genre, price, stock
2. customers.csv – name, city, country
3. orders.csv – book ID, customer ID, quantity, total amount, order date

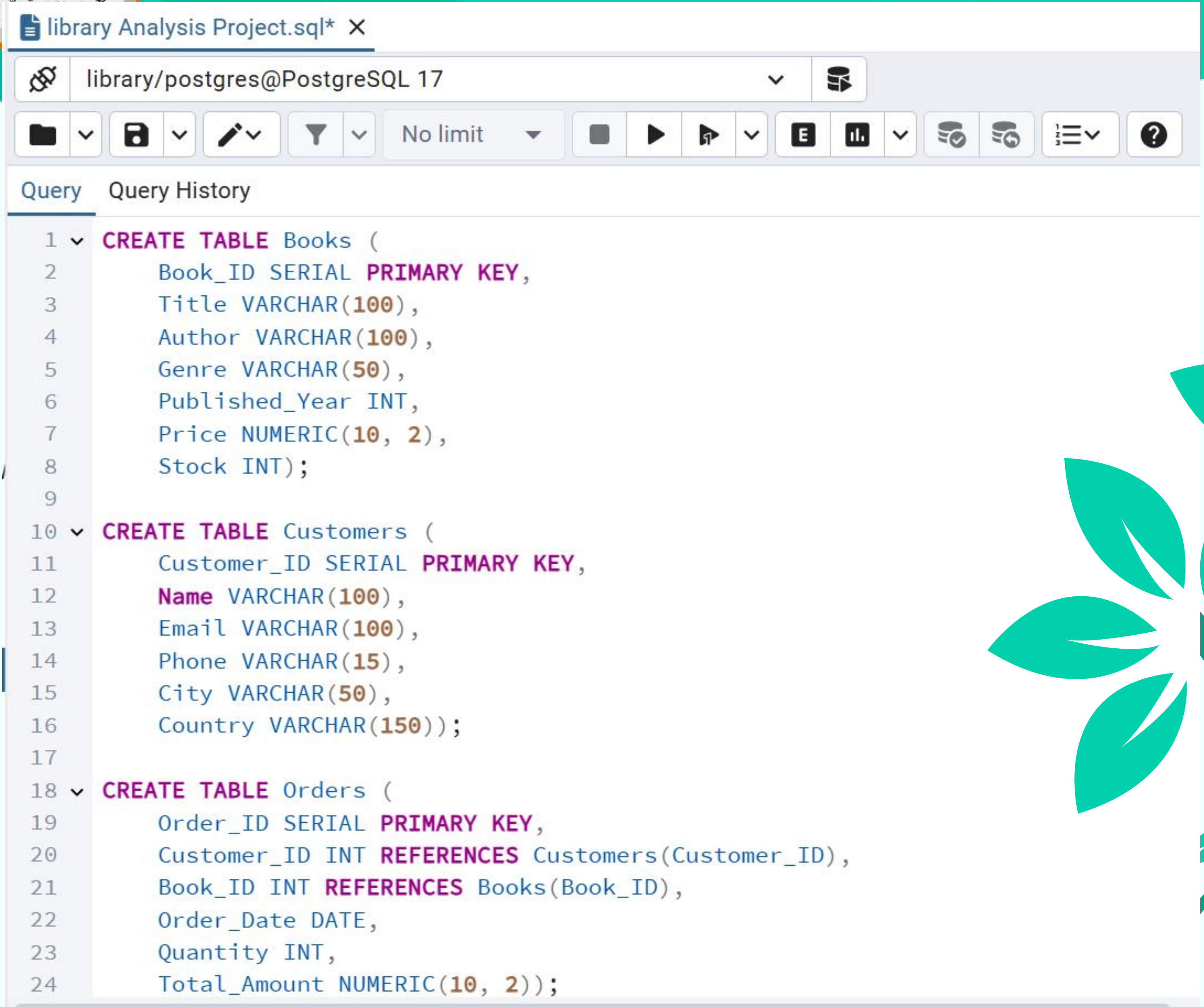
# Datasets Used

## 3 CSV Files

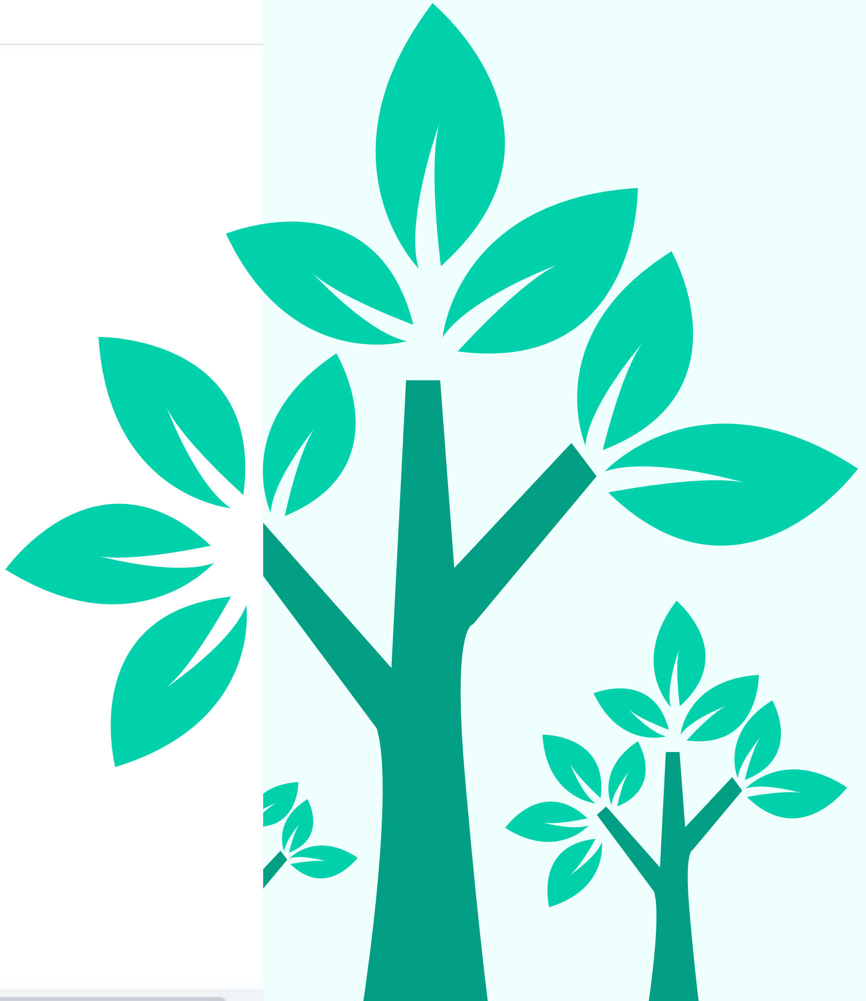
Tables must have at least one common column with same column name and same data type



# Sample Table

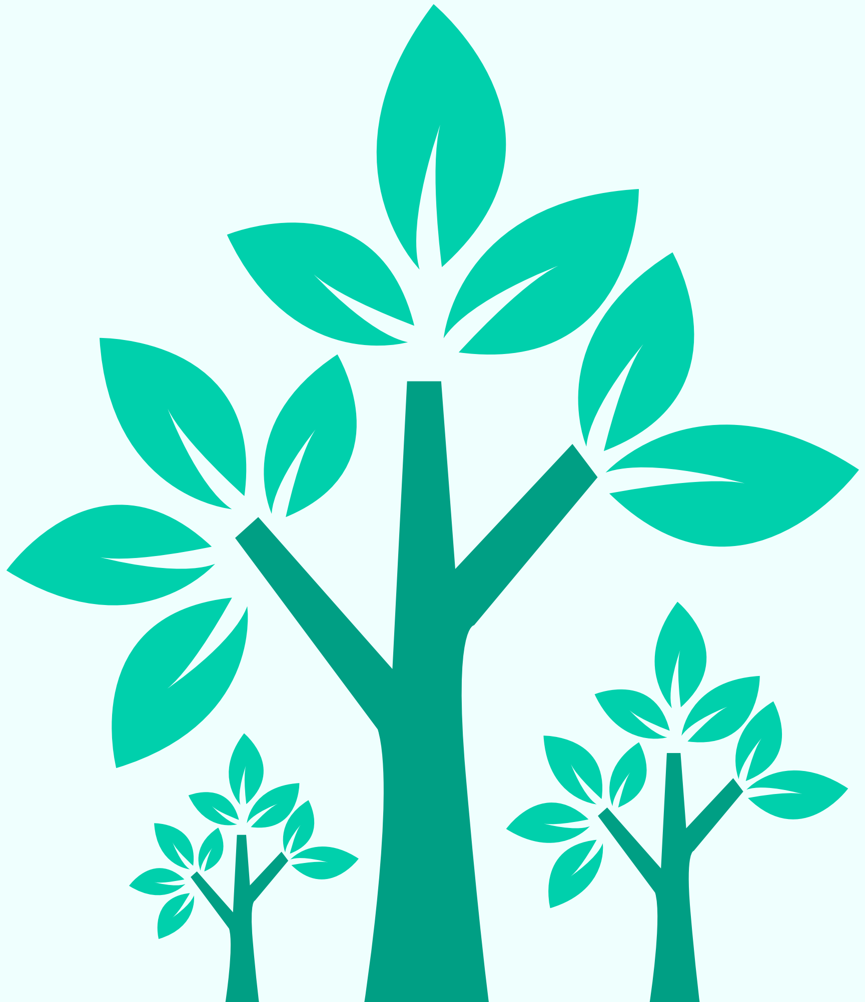


```
CREATE TABLE Books (
	Book_ID SERIAL PRIMARY KEY,
	Title VARCHAR(100),
	Author VARCHAR(100),
	Genre VARCHAR(50),
	Published_Year INT,
	Price NUMERIC(10, 2),
	Stock INT);

CREATE TABLE Customers (
	Customer_ID SERIAL PRIMARY KEY,
	Name VARCHAR(100),
	Email VARCHAR(100),
	Phone VARCHAR(15),
	City VARCHAR(50),
	Country VARCHAR(150));

CREATE TABLE Orders (
	Order_ID SERIAL PRIMARY KEY,
	Customer_ID INT REFERENCES Customers(Customer_ID),
	Book_ID INT REFERENCES Books(Book_ID),
	Order_Date DATE,
	Quantity INT,
	Total_Amount NUMERIC(10, 2));
```



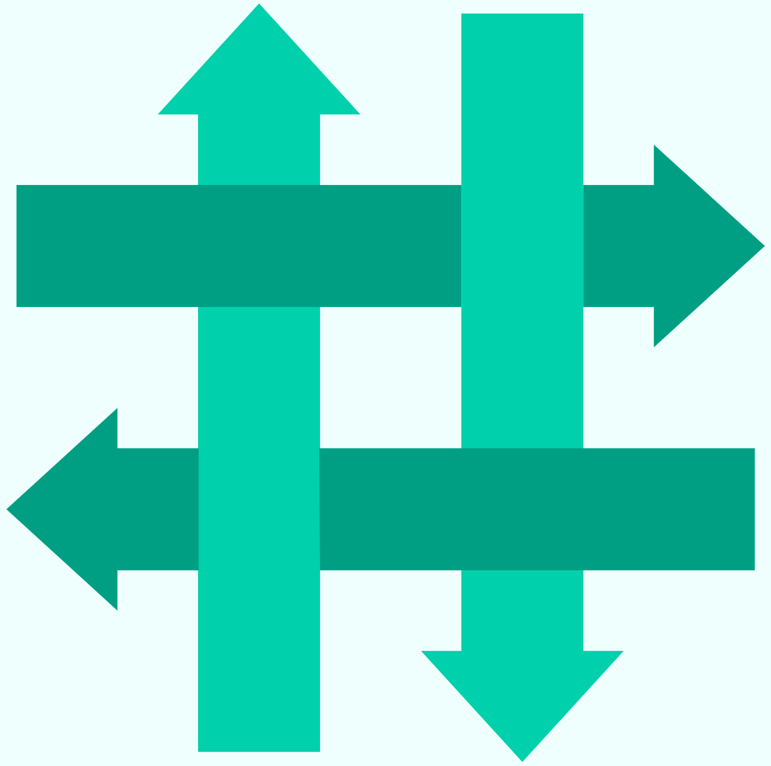
# Objectives

- Analyze book sales performance
- Understand customer behavior
- Monitor inventory levels
- Calculate total revenue
- Identify top authors and regions



# Business Problems

- Best performing genres/books
- Total revenue



- Most loyal and profitable customers
- Stock after orders
- Top cities/regions by value

# Methodology

Import and clean CSV data

Basic SQL exploration

Advanced SQL with JOINs & aggregations

KPI modeling

Generate insights

## Key Insights

- Fiction: highest-selling genre
- High-spending customers identified
- Total revenue calculated
- Remaining stock determined
- Top authors/books listed

# Skills Gained

- Real-world SQL queries

- Managing relational tables

- Translating KPIs to database logic

- Business-focused data analysis mindset

## Basic Queries

1) Retrieve all books in the "Fiction" genre

2) Find books published after the year 1950

3) List all customers from the Canada

4) Show orders placed in November 2023

5) Retrieve the total stock of books available

6) Find the details of the most expensive book

7) Show all customers who ordered more than 1 quantity of a book

8) Retrieve all orders where the total amount exceeds $20

9) List all genres available in the Books table

10) Find the book with the lowest stock

11) Calculate the total revenue generated from all orders

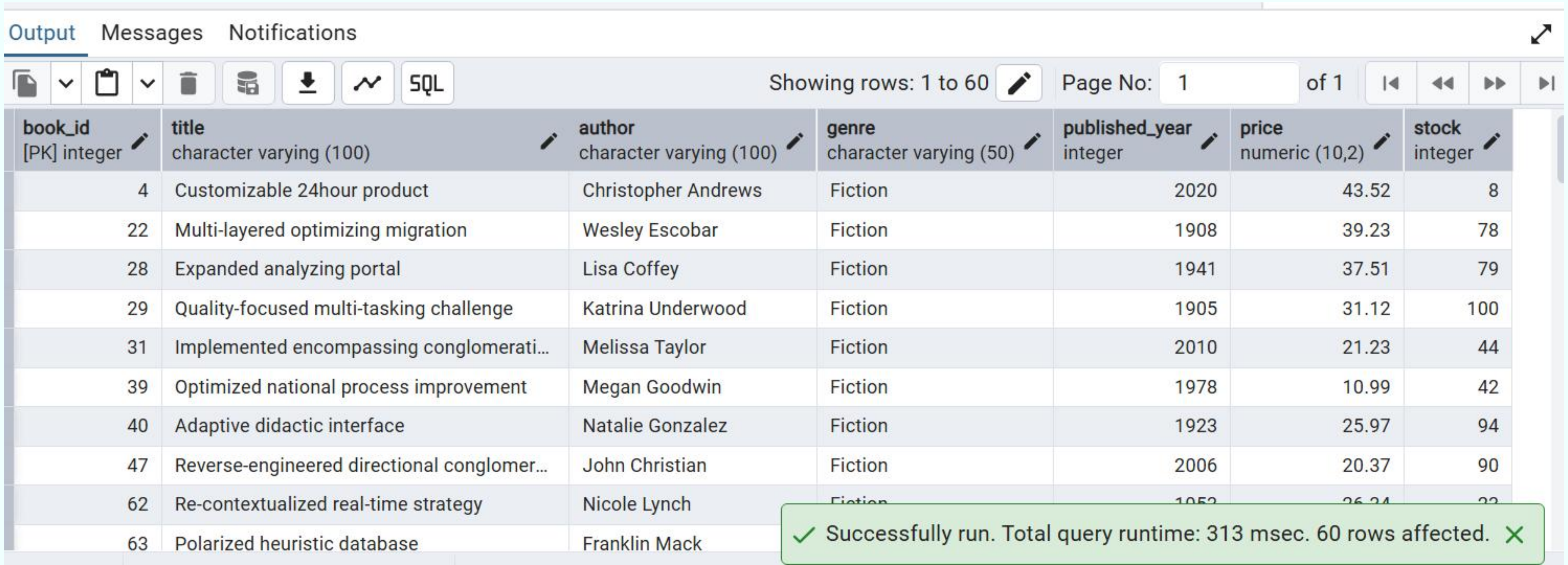
## Advance Queries

1) Retrieve the total number of books sold for each genre
2) Find the average price of books in the "Fantasy" genre
3) List customers who have placed at least 2 orders
4) Find the most frequently ordered book
5) Show the top 3 most expensive books of 'Fantasy' Genre
6) Retrieve the total quantity of books sold by each author
7) List the cities where customers who spent over $30 are located
8) Find the customer who spent the most on orders
9) Calculate the stock remaining after fulfilling all orders

# Basic Queries

1.Retrieve all books in the “Fiction”genre.

```
SELECT*
FROM books
WHERE genre ='Fiction';
```



Showing rows: 1 to 60 Page No: 1 of 1

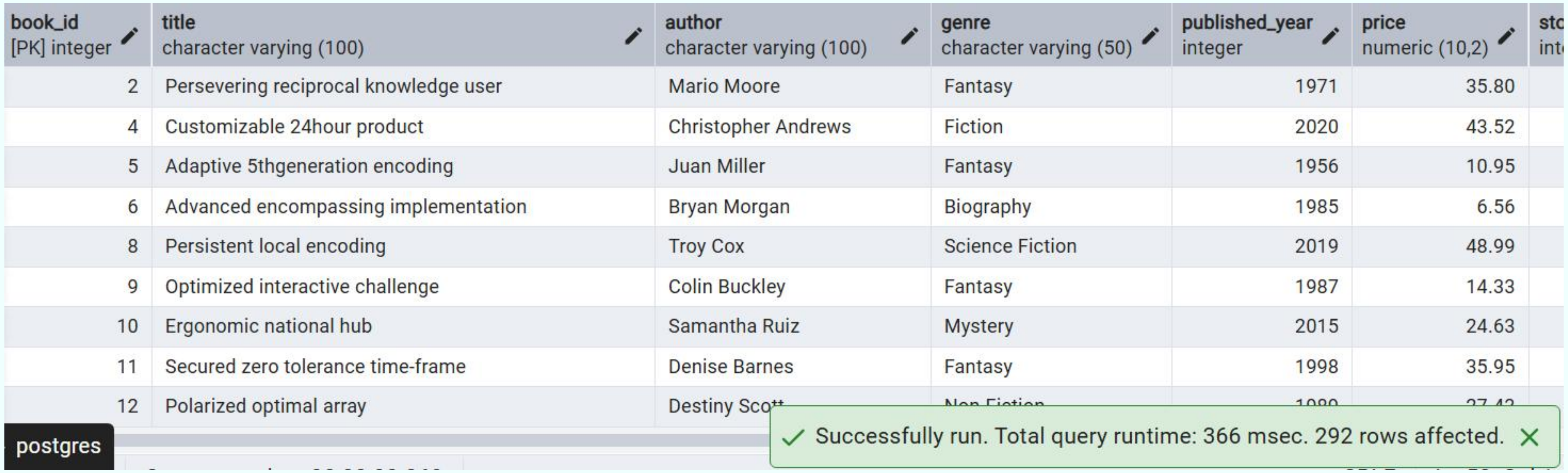
| book_id [PK] integer | title character varying (100) | author character varying (100) | genre character varying (50) | published_year integer | price numeric (10,2) | stock integer |
|---|---|---|---|---|---|---|
| 4 | Customizable 24hour product | Christopher Andrews | Fiction | 2020 | 43.52 | 8 |
| 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 conglomerati... | 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 conglomer... | John Christian | Fiction | 2006 | 20.37 | 90 |
| 62 | Re-contextualized real-time strategy | Nicole Lynch | Fiction | 1953 | 26.24 | 22 |
| 63 | Polarized heuristic database | Franklin Mack | | | | |

✓ Successfully run. Total query runtime: 313 msec. 60 rows affected.

# Basic Queries

2) Find books published after the year 1950.

```
SELECT *
FROM Books
WHERE published_year > '1950';
```



| book_id [PK] integer | title character varying (100) | author character varying (100) | genre character varying (50) | published_year integer | price numeric (10,2) | sto int |
|---|---|---|---|---|---|---|
| 2 | Persevering reciprocal knowledge user | Mario Moore | Fantasy | 1971 | 35.80 | |
| 4 | Customizable 24hour product | Christopher Andrews | Fiction | 2020 | 43.52 | |
| 5 | Adaptive 5thgeneration encoding | Juan Miller | Fantasy | 1956 | 10.95 | |
| 6 | Advanced encompassing implementation | Bryan Morgan | Biography | 1985 | 6.56 | |
| 8 | Persistent local encoding | Troy Cox | Science Fiction | 2019 | 48.99 | |
| 9 | Optimized interactive challenge | Colin Buckley | Fantasy | 1987 | 14.33 | |
| 10 | Ergonomic national hub | Samantha Ruiz | Mystery | 2015 | 24.63 | |
| 11 | Secured zero tolerance time-frame | Denise Barnes | Fantasy | 1998 | 35.95 | |
| 12 | Polarized optimal array | Destiny Scott | Non Fiction | [illegible] | 27.42 | |

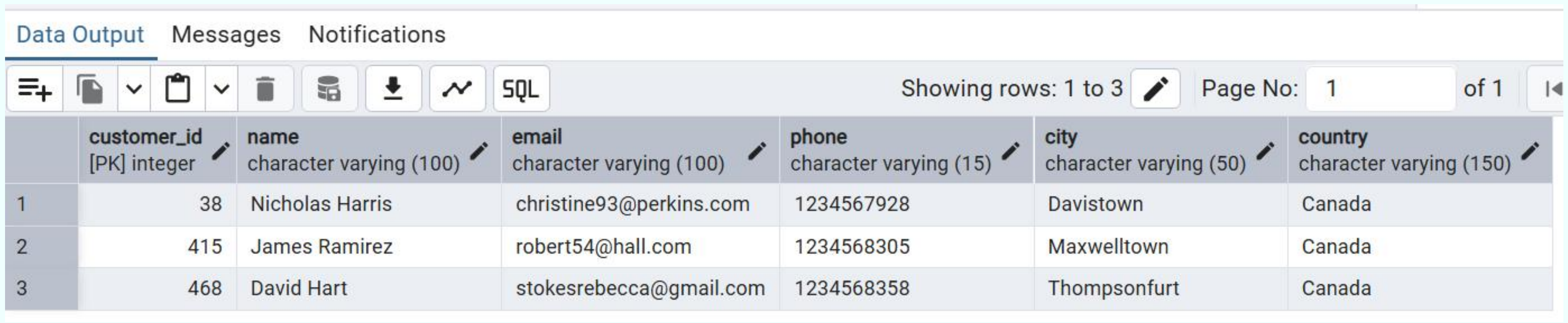
postgres

✓ Successfully run. Total query runtime: 366 msec. 292 rows affected. ✕

# Basic Queries

3) List all customers from the Canada.

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

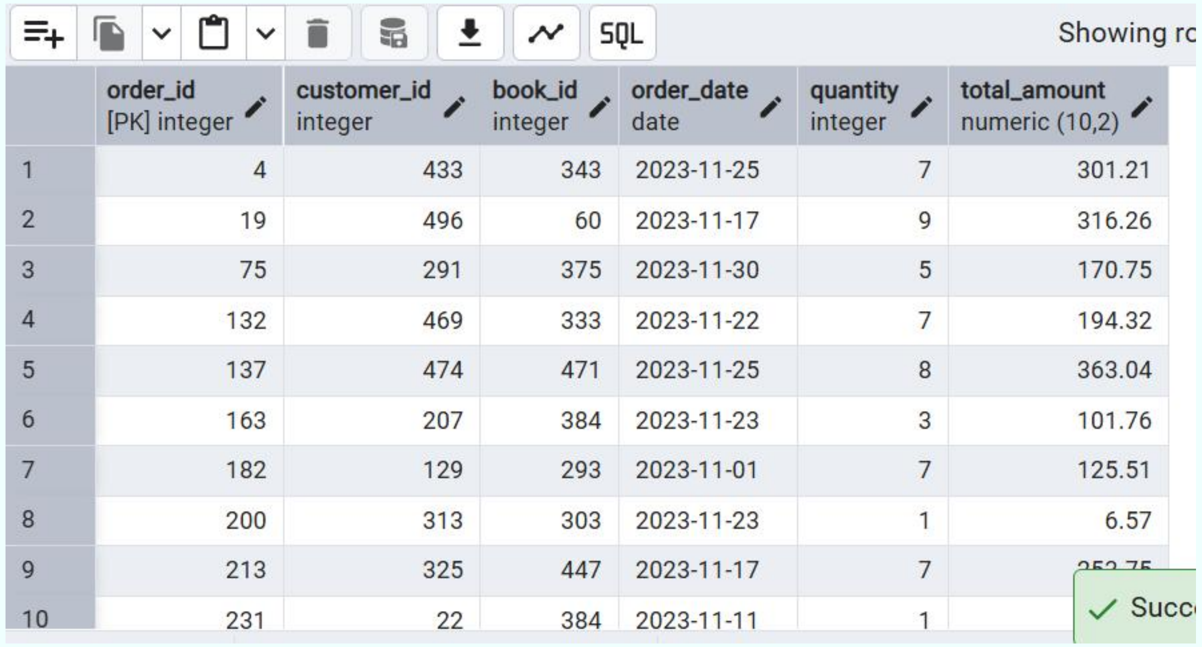


| | customer_id [PK] integer | name character varying (100) | email character varying (100) | phone character varying (15) | city character varying (50) | country character varying (150) |
|---|---|---|---|---|---|---|
| 1 | 38 | Nicholas Harris | christine93@perkins.com | 1234567928 | Davistown | Canada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

# Basic Queries

4) Show orders placed in November 2023.

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

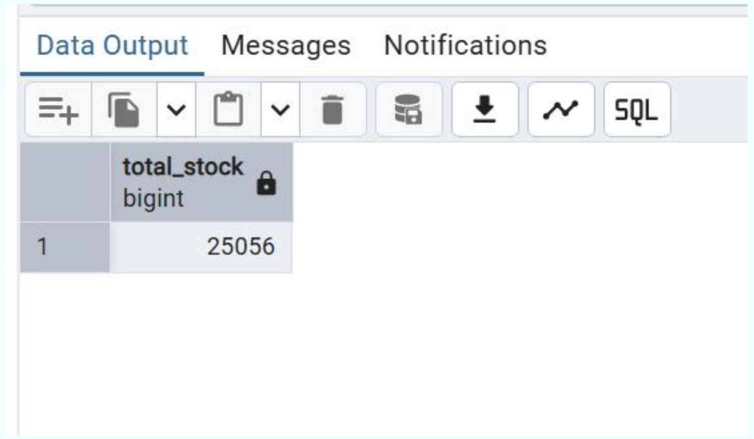


| | order_id [PK] integer | customer_id integer | book_id integer | order_date date | quantity integer | total_amount numeric (10,2) |
|---|---|---|---|---|---|---|
| 1 | 4 | 433 | 343 | 2023-11-25 | 7 | 301.21 |
| 2 | 19 | 496 | 60 | 2023-11-17 | 9 | 316.26 |
| 3 | 75 | 291 | 375 | 2023-11-30 | 5 | 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 | 252.75 |
| 10 | 231 | 22 | 384 | 2023-11-11 | 1 | [illegible] |

# Basic Queries

5) Retrieve the total stock of books available.

```
SELECT
SUM(stock) AS
total_stock
FROM Books;
```

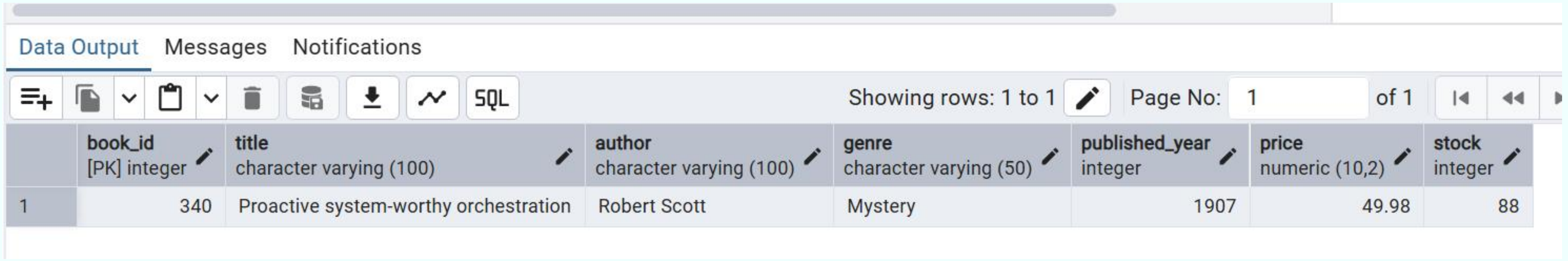


```
   total_stock
   bigint
1        25056
```

# Basic Queries

6) Find the details of the most expensive book.

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

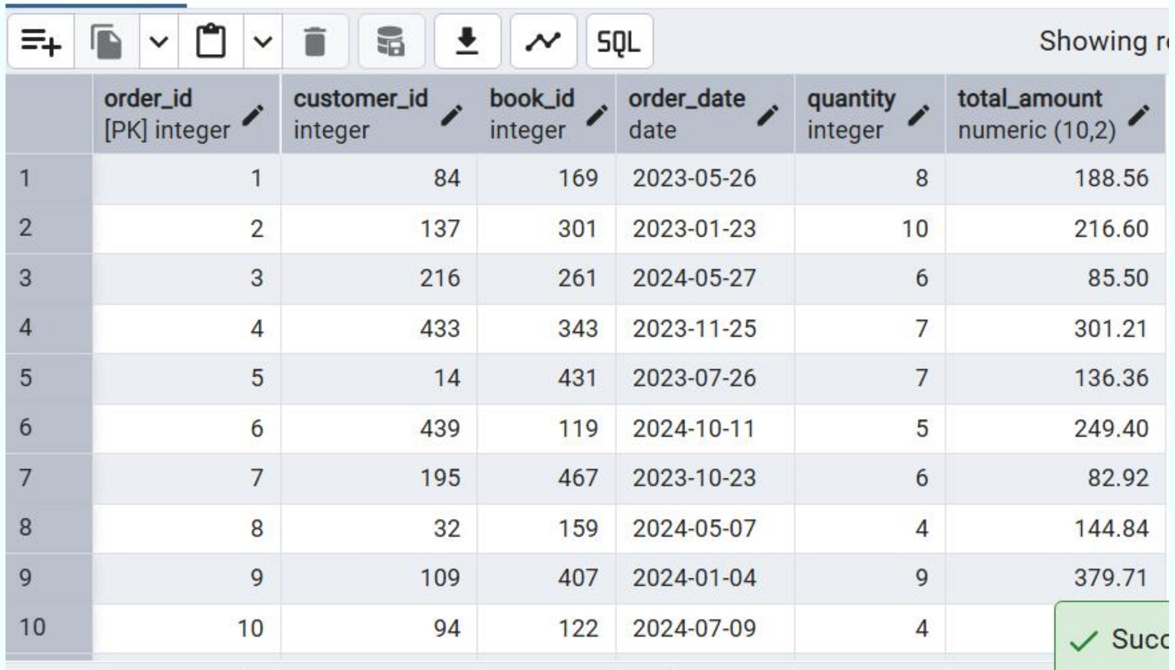


| | book_id [PK] integer | title character varying (100) | author character varying (100) | genre character varying (50) | published_year integer | price numeric (10,2) | stock integer |
|---|---|---|---|---|---|---|---|
| 1 | 340 | Proactive system-worthy orchestration | Robert Scott | Mystery | 1907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

# Basic Queries

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

```
SELECT *
FROM Orders
WHERE quantity > 1;
```

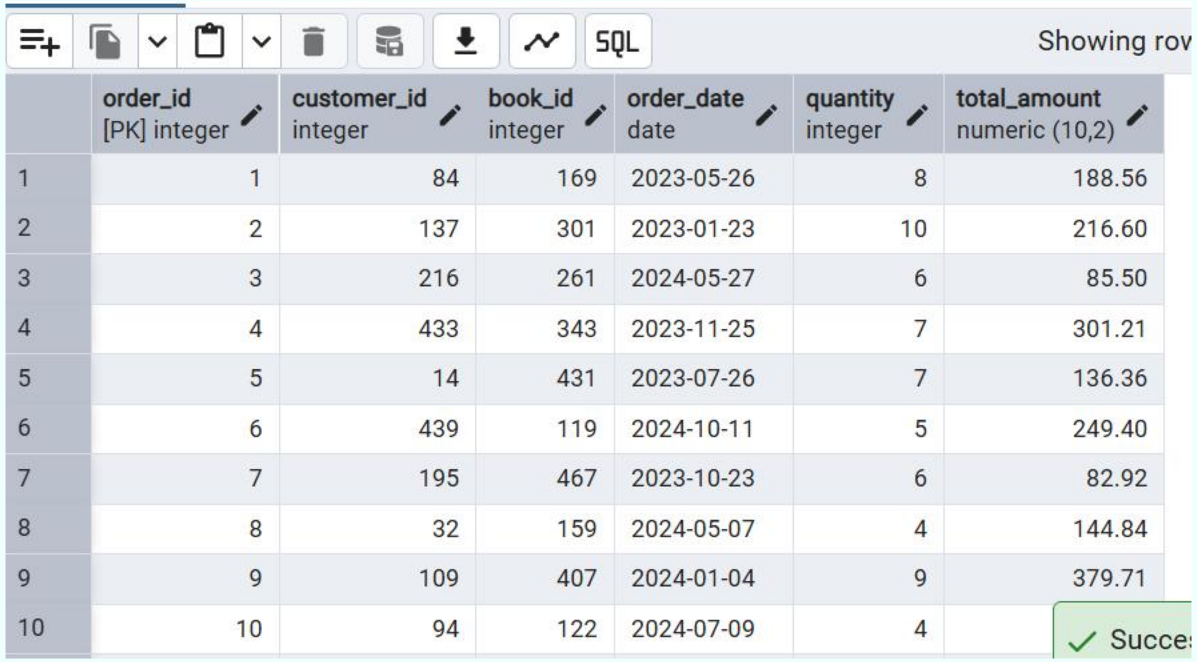


| | order_id [PK] integer | customer_id integer | book_id integer | order_date date | quantity integer | total_amount numeric (10,2) |
|---|---|---|---|---|---|---|
| 1 | 1 | 84 | 169 | 2023-05-26 | 8 | 188.56 |
| 2 | 2 | 137 | 301 | 2023-01-23 | 10 | 216.60 |
| 3 | 3 | 216 | 261 | 2024-05-27 | 6 | 85.50 |
| 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 | |

# Basic Queries

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

```
SELECT *
FROM Orders
WHERE total_amount > 20;
```

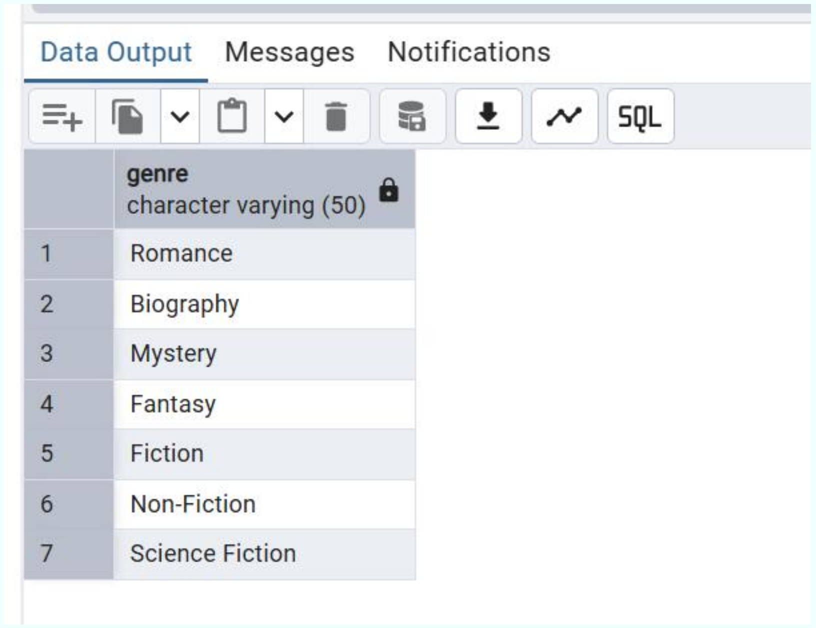


| | order_id [PK] integer | customer_id integer | book_id integer | order_date date | quantity integer | total_amount numeric (10,2) |
|---|---|---|---|---|---|---|
| 1 | 1 | 84 | 169 | 2023-05-26 | 8 | 188.56 |
| 2 | 2 | 137 | 301 | 2023-01-23 | 10 | 216.60 |
| 3 | 3 | 216 | 261 | 2024-05-27 | 6 | 85.50 |
| 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 | |

# Basic Queries

9) List all genres available in the Books table.

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

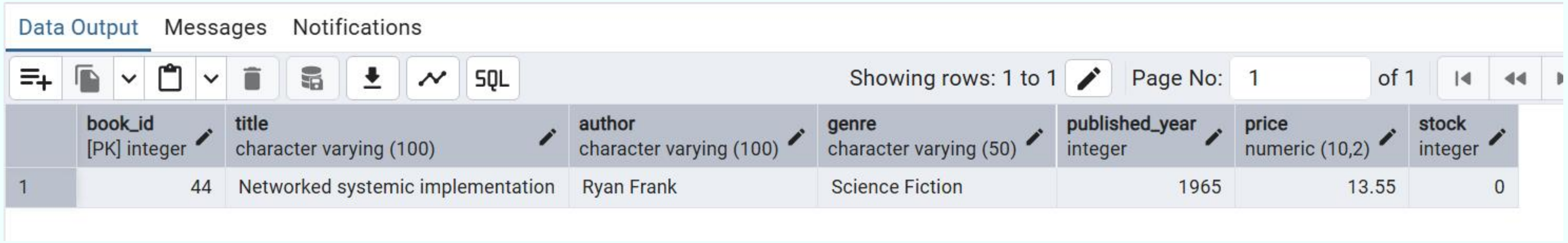


|   | genre character varying (50) |
|---|---|
| 1 | Romance |
| 2 | Biography |
| 3 | Mystery |
| 4 | Fantasy |
| 5 | Fiction |
| 6 | Non-Fiction |
| 7 | Science Fiction |

# Basic Queries

10) Find the book with the lowest stock.

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

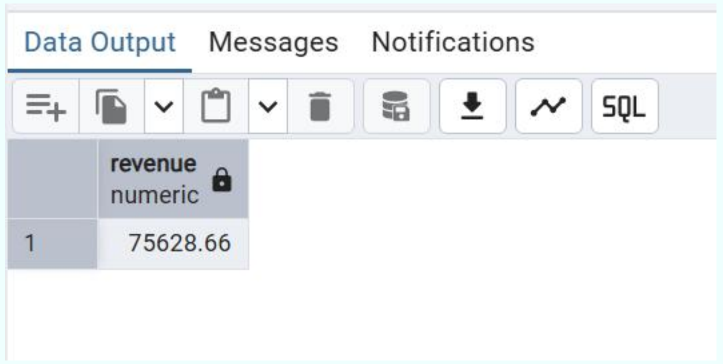


| | book_id [PK] integer | title character varying (100) | author character varying (100) | genre character varying (50) | published_year integer | price numeric (10,2) | stock integer |
|---|---|---|---|---|---|---|---|
| 1 | 44 | Networked systemic implementation | Ryan Frank | Science Fiction | 1965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

# Basic Queries

11) Calculate the total revenue generated from all orders.

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

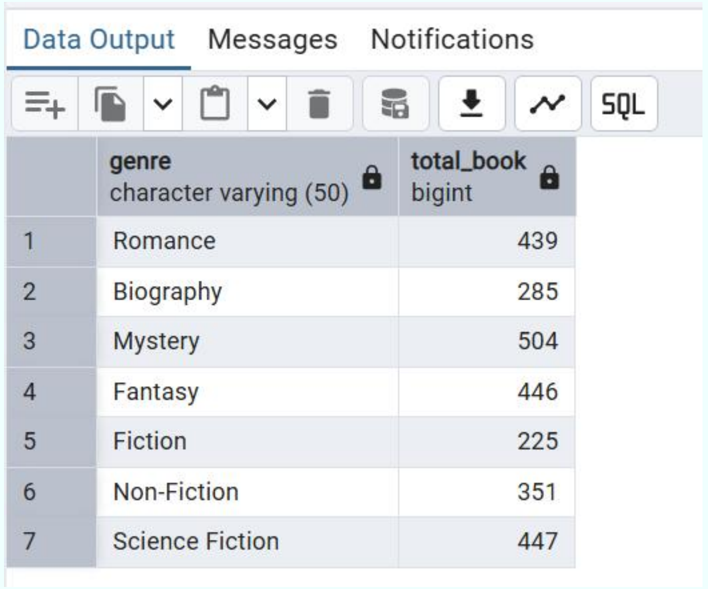


```
   revenue
   numeric
1  75628.66
```

# Advance Queries

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

```
SELECT b.genre,
SUM(o.quantity) AS total_book
FROM orders O
join books b
ON o.book_id = b.book_id
GROUP BY b.genre;
```

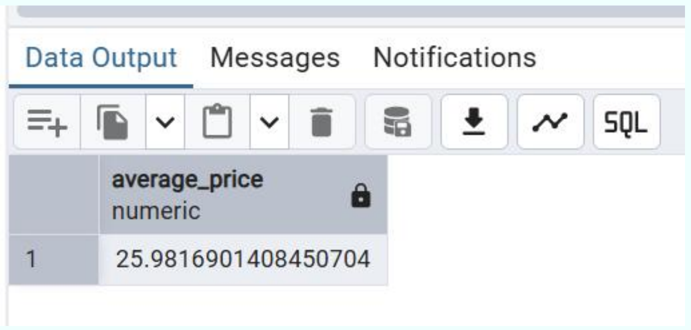


| | genre character varying (50) | total_book bigint |
|---|---|---|
| 1 | Romance | 439 |
| 2 | Biography | 285 |
| 3 | Mystery | 504 |
| 4 | Fantasy | 446 |
| 5 | Fiction | 225 |
| 6 | Non-Fiction | 351 |
| 7 | Science Fiction | 447 |

# Advance Queries

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

```
SELECT
AVG(price) AS average_price
FROM Books
WHERE genre = 'Fantasy'
GROUP BY genre;
```

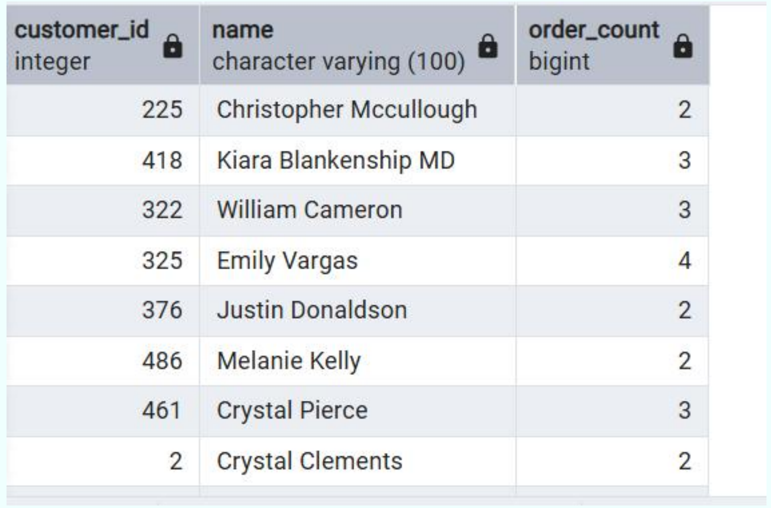


```
   average_price
   numeric
1  25.9816901408450704
```

# Advance Queries

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;
```

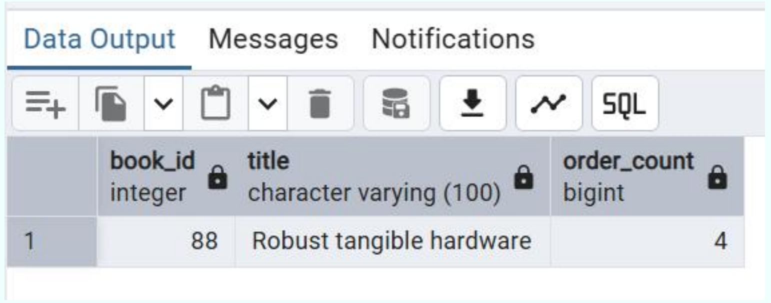


| customer_id integer | name character varying (100) | order_count bigint |
|---|---|---|
| 225 | Christopher Mccullough | 2 |
| 418 | Kiara Blankenship MD | 3 |
| 322 | William Cameron | 3 |
| 325 | Emily Vargas | 4 |
| 376 | Justin Donaldson | 2 |
| 486 | Melanie Kelly | 2 |
| 461 | Crystal Pierce | 3 |
| 2 | Crystal Clements | 2 |

# Advance Queries

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;
```

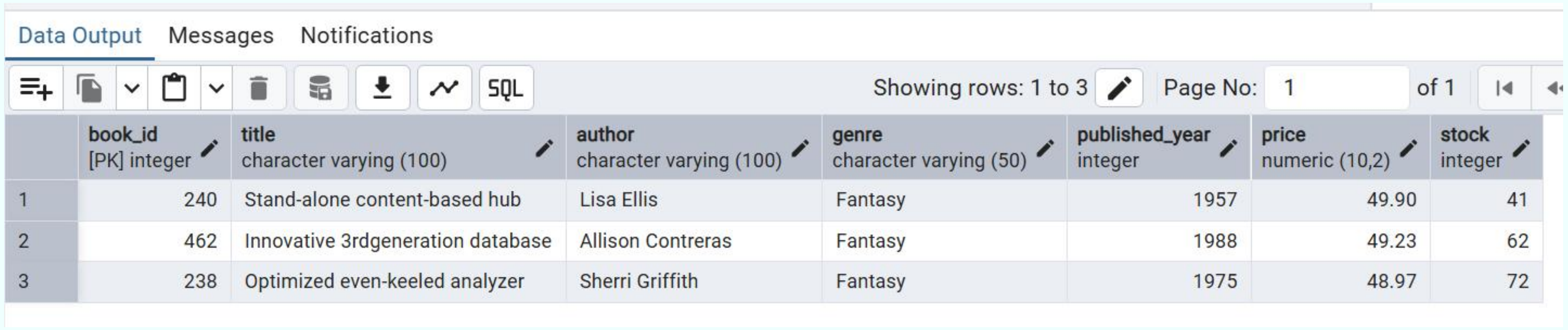


| | book_id integer | title character varying (100) | order_count bigint |
|---|---|---|---|
| 1 | 88 | Robust tangible hardware | 4 |

# Advance Queries

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

```
SELECT *
FROM books
WHERE  genre = 'Fantasy'
ORDER BY price DESC
LIMIT 3;
```

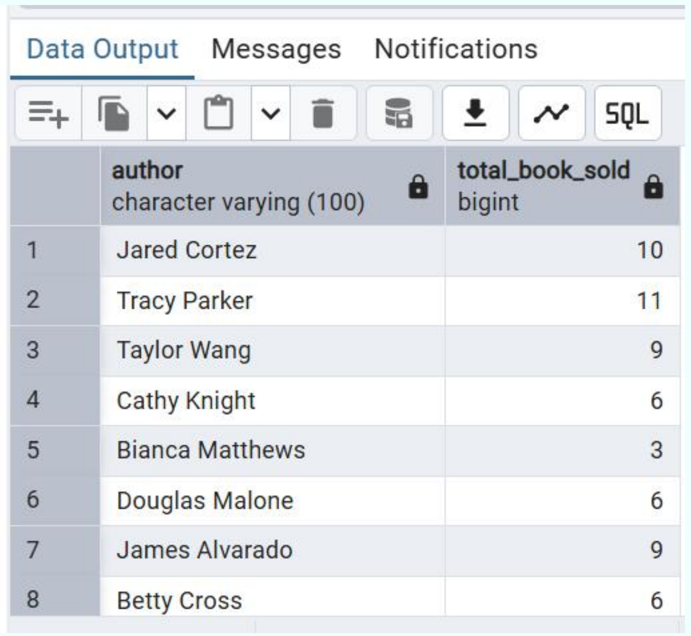


| | book_id [PK] integer | title character varying (100) | author character varying (100) | genre character varying (50) | published_year integer | price numeric (10,2) | stock integer |
|---|---|---|---|---|---|---|---|
| 1 | 240 | Stand-alone content-based hub | Lisa Ellis | Fantasy | 1957 | 49.90 | 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 |

# Advance Queries

17) Retrieve the total quantity of books sold by each author.

```
SELECT b.author,
SUM(o.quantity) AS total_book_sold
FROM orders o
JOIN books b
ON b.book_id = o.book_id
GROUP BY b.author;
```

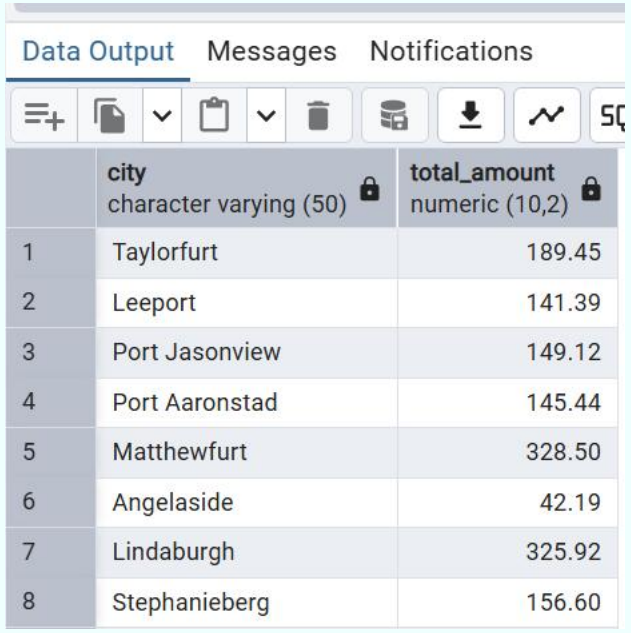


| | author character varying (100) | total_book_sold bigint |
|---|---|---|
| 1 | Jared Cortez | 10 |
| 2 | Tracy Parker | 11 |
| 3 | Taylor Wang | 9 |
| 4 | Cathy Knight | 6 |
| 5 | Bianca Matthews | 3 |
| 6 | Douglas Malone | 6 |
| 7 | James Alvarado | 9 |
| 8 | Betty Cross | 6 |

# Advance Queries

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;
```

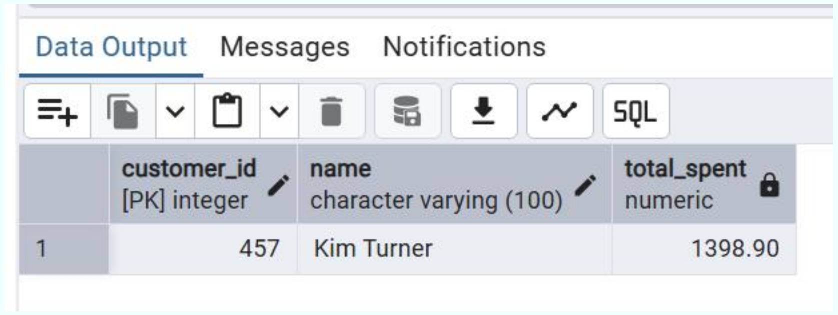


| | city character varying (50) | total_amount numeric (10,2) |
|---|---|---|
| 1 | Taylorfurt | 189.45 |
| 2 | Leeport | 141.39 |
| 3 | Port Jasonview | 149.12 |
| 4 | Port Aaronstad | 145.44 |
| 5 | Matthewfurt | 328.50 |
| 6 | Angelaside | 42.19 |
| 7 | Lindaburgh | 325.92 |
| 8 | Stephanieberg | 156.60 |

# Advance Queries

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 1;
```

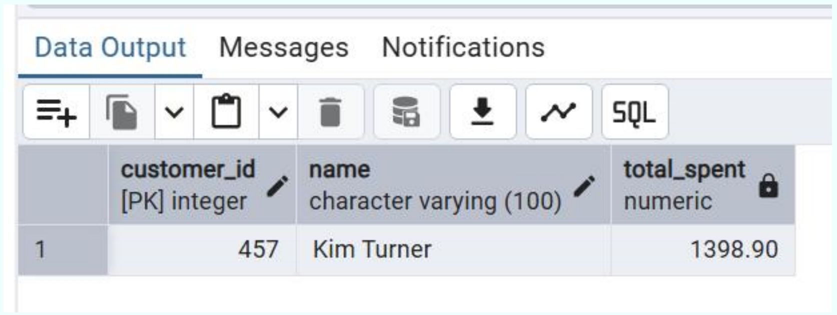


| | customer_id [PK] integer | name character varying (100) | total_spent numeric |
|---|---|---|---|
| 1 | 457 | Kim Turner | 1398.90 |

# Advance Queries

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;
```



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

# Conclusion

1

First complete SQL project - from CSVs to KPIs.

2

Improved SQL skills and business analysis thinking.

3

Strong portfolio addition for GitHub and LinkedIn.

# Thank You!

Every great presentation is complete with a great audience — and that's you!