Online BookStore SQL Project

RELATIONAL DATABASE DESIGN & ANALYTICS USING POSTGRESQL

By:
Abdul Razzaq

Project Overview

- Designed and implemented a relational database for an Online Bookstore using PostgreSQL.
- Created three main tables: Books, Customers, and Orders with appropriate relationships.
- Imported real-world style data from CSV files using the COPY command.
- Wrote analytical SQL queries to extract actionable business insights.
- Applied core SQL concepts: joins, groupings, aggregations, subqueries, and window functions.
- Focused on performance metrics such as total revenue, customer spending, and product sales.
- The project simulates real-life business reporting and inventory analysis.

Database & Table Creation (DDL)

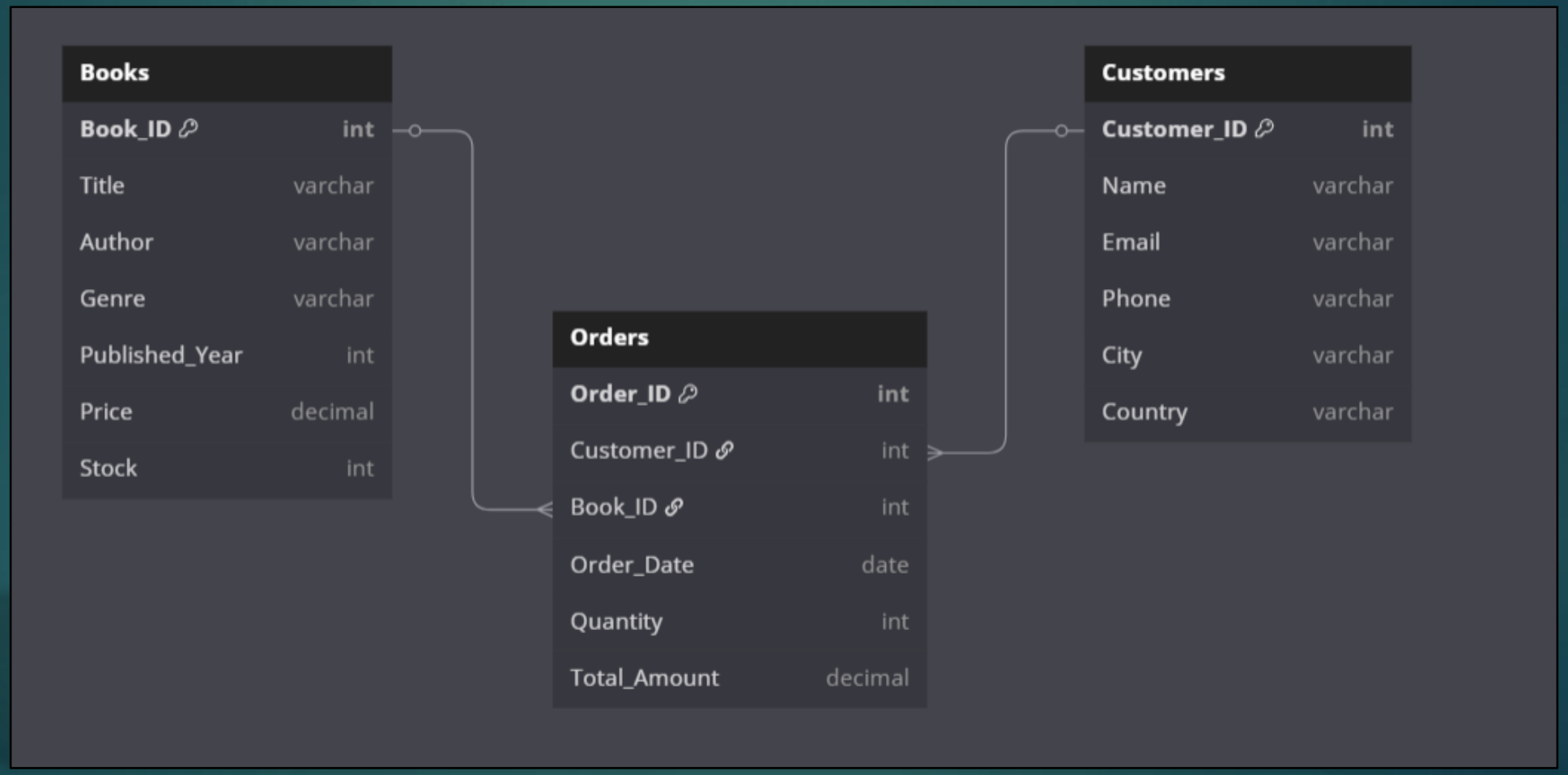
- Created OnlineBookstore database using PostgreSQL.
- Defined three tables: Books, Customers, and Orders.
- Applied Primary Keys and Foreign Key constraints for relationships.
- Structured schema to support clean, normalized data flow.

Query

```sql
-- Create Database
CREATE DATABASE OnlineBookstore;
```

```sql
-- Create Books 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 Customers Table
CREATE TABLE Customers (
    Customer_ID SERIAL PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(15),
    City VARCHAR(50),
    Country VARCHAR(150)
);
-- Create Orders Table
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)
);
```

ER Diagram


Books
Book_ID int
Title varchar
Author varchar
Genre varchar
Published_Year int
Price decimal
Stock int
Orders
Order_ID int
Customer_ID int
Book_ID int
Order_Date date
Quantity int
Total_Amount decimal
Customers
Customer_ID int
Name varchar
Email varchar
Phone varchar
City varchar
Country varchar

CSV Data Import into Tables

- Imported real-world-style data using PostgreSQL's COPY command.
- Used .csv files for Books, Customers, and Orders tables.
- Ensured headers matched table columns to avoid import errors.
- Data import enabled meaningful analysis on sales, inventory, and customer behavior.

Query

```sql
-- Import Data into Books Table
COPY Books(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)
FROM 'D:\SQL\CSV\Books.csv'
CSV HEADER;

-- Import Data into Customers Table
COPY Customers(Customer_ID, Name, Email, Phone, City, Country)
FROM 'D:\SQL\CSV\Customers.csv'
CSV HEADER;

-- Import Data into Orders Table
COPY Orders(Order_ID, Customer_ID, Book_ID, Order_Date, Quantity, Total_Amount)
FROM 'D:\SQL\CSV\Orders.csv'
CSV HEADER;
```

SQL Query Analysis Goals

Key Points

- Uncover insights to drive bookstore decisions.
- Focus: Customer behavior, inventory, revenue, genre trends.
- Use advanced SQL for actionable results.
- Impact: Better marketing, optimized stock, higher profits.

Customer Order Frequency Analysis

Insight

This query counts the total number of orders placed by each customer. It helps identify how often individual customers interact with the bookstore and shows overall engagement levels.

Query

```sql
-- Find the total number of orders per customer
SELECT
    customer_id,
    COUNT(order_id) AS total_orders
FROM
    orders
GROUP BY
    customer_id
ORDER BY
    total_orders DESC;
```

Query Output

	customer_id integer	total_orders bigint
1	364	6
2	474	5
3	405	4
4	485	4
5	437	4
6	425	4
7	107	4
8	174	4
9	457	4
10	325	4

Top 5 Most Expensive Books

Insight

This query selects the five most expensive books by sorting the Books table in descending order of price. It highlights premium titles that may require special marketing or attention.

Query

```sql
-- Retrieve top 5 most expensive books
SELECT
    *
FROM
    books
ORDER BY
    price DESC
LIMIT
    5;
```

Query Output

	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 standardiza...	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

Customers With No Orders

Insight

This query finds customers who have never placed an order by checking which customer IDs do not appear in the Orders table. It helps spot inactive users in the system.

Query

```sql
-- List customers who have not placed any orders
SELECT
    customer_id,
    name,
    country
FROM
    customers
WHERE
    customer_id NOT IN (
        SELECT
            customer_id
        FROM
            orders
    );
```

Query Output

	customer_id [PK] integer	name character varying (100)	country character varying (150)
1	1	Deborah Griffith	Denmark
2	3	Susan Fuller	Equatorial Guinea
3	4	Jamie Ramirez	Slovenia
4	5	Marcus Murphy	Guinea-Bissau
5	9	Matthew Williams	Somalia
6	19	Marc Nash	French Guiana
7	20	Robert Salas	Denmark
8	25	James Martin	Yemen
9	28	William Burns	Netherlands
10	29	John Maxwell	Cuba

Comprehensive Order Details

Insight

This query joins the Orders, Customers, and Books tables to display full order records, including customer names and book titles. It's useful for creating reports or detailed order views.

Query

```sql
-- Show all orders with customer name and book title
SELECT
    o.order_id,
    o.order_date,
    o.quantity,
    o.total_amount,
    c.name AS customer_name,
    b.title AS book_title
FROM
    orders o
JOIN customers c ON c.customer_id = o.customer_id
JOIN books b ON b.book_id = o.book_id;
```

Query Output

	order_id integer	order_date date	quantity integer	total_amount numeric (10,2)	customer_name character varying (100)	book_title character varying (100)
1	1	2023-05-26	8	188.56	Gary Blair	Networked tertiary approach
2	2	2023-01-23	10	216.60	Steven Miller	Polarized high-level installation
3	3	2024-05-27	6	85.50	Phillip Allen	Intuitive content-based toolset
4	4	2023-11-25	7	301.21	Corey Wells	De-engineered grid-enabled secured line
5	5	2023-07-26	7	136.36	John Wood	Synergized fresh-thinking monitoring
6	6	2024-10-11	5	249.40	Shane Chang	Switchable modular moratorium
7	7	2023-10-23	6	82.92	Dominique Turner	Function-based dedicated frame
8	8	2024-05-07	4	144.84	Jeffrey Shannon	Proactive 5thgeneration middleware
9	9	2024-01-04	9	379.71	Jacob Kelley	Mandatory executive groupware
10	10	2024-07-09	4	123.00	Mr. David Cox	Profound leadingedge capability

Identifying Science Fiction Book Buyers

Insight

This query filters orders to return only those linked to books from the "Science Fiction" genre. It also includes customer details, helping analyze genre-specific customer interest and behavior.

Query

```sql
-- Customers who ordered Sci-Fi books
SELECT
    o.order_id,
    o.book_id,
    c.customer_id,
    c.name,
    c.country,
    b.genre
FROM
    orders   o
JOIN
    customers c ON c.customer_id = o.customer_id
JOIN
    books     b ON b.book_id    = o.book_id
WHERE
    b.genre = 'Science Fiction';
```

Query Output

	order_id integer	book_id integer	customer_id integer	name character varying (100)	country character varying (150)	genre character varying (50)
1	244	380	12	Jennifer Murray	Zimbabwe	Science Fiction
2	435	230	13	Kristine Kim	Nigeria	Science Fiction
3	414	234	23	Hannah Drake	Faroe Islands	Science Fiction
4	412	103	24	Christina Mitchell	Trinidad and Tobago	Science Fiction
5	332	113	41	Samuel Daniels	Benin	Science Fiction
6	78	196	44	Alexis Gallegos	Lesotho	Science Fiction
7	250	65	63	Manuel Lewis	Heard Island and McDonald Islands	Science Fiction
8	113	255	80	Joy Fisher	Tuvalu	Science Fiction
9	229	482	82	Shaun Fernandez	Malaysia	Science Fiction
10	126	492	88	Gregory Joseph	Algeria	Science Fiction

Overall Revenue Snapshot

Insight

This query calculates the total revenue generated by summing the Total_Amount column across all orders. It provides a quick and accurate overview of overall business sales performance

Query

```sql
-- Total revenue from all orders
SELECT
    SUM(total_amount) AS total_revenue
FROM
    orders;
```

Query Output

	total_revenue numeric
1	75628.66

Customer Spending Profiles

Insight

This query returns the name, email, and total spending amount for each customer. It helps identify top spenders and analyze purchase behavior for customer-level financial insights.

Query

```sql
-- Customer name, email, and total spending
SELECT
    c.customer_id,
    c.name,
    c.email,
    SUM(o.total_amount) AS total_spent
FROM
    orders o
JOIN
    customers c ON c.customer_id = o.customer_id
GROUP BY
    c.customer_id,
    c.name,
    c.email
ORDER BY
    total_spent DESC;
```

Query Output

	customer_id [PK] integer	name character varying (100)	email character varying (100)	total_spent numeric
1	457	Kim Turner	jennifer45@weiss-perry.com	1398.90
2	174	Jonathon Strickland	ryan10@yahoo.com	1080.95
3	364	Carrie Perez	chelsea23@gillespie-walker.com	1052.27
4	405	Julie Smith	knightmonica@krueger-hamilton.biz	991.00
5	386	Pamela Gordon	mandy28@thomas-white.com	986.30
6	425	Ashley Perez	williamslindsey@yahoo.com	942.62
7	474	Anthony Young	rogersbill@gmail.com	929.19
8	163	Robert Clark	sheilalester@gmail.com	746.65
9	167	Justin Spencer	michaelsnyder@gmail.com	719.93
10	214	Alexander Scott	amypierce@hotmail.com	682.15

Unsold Books Inventory

Insight

This query identifies books from the Books table that have never been ordered by checking which book IDs do not exist in the Orders table. It helps detect unsold inventory.

Query

```sql
-- Books that were never ordered
SELECT
    book_id,
    title
FROM
    books
WHERE
    book_id NOT IN (
        SELECT
            book_id
        FROM
            orders
    );
```

Query Output

	book_id [PK] integer	title character varying (100)
1	2	Persevering reciprocal knowledge user
2	4	Customizable 24hour product
3	6	Advanced encompassing implementation
4	9	Optimized interactive challenge
5	12	Polarized optimal array
6	14	Re-engineered demand-driven parallelism
7	15	User-friendly motivating strategy
8	20	Face-to-face systematic throughput
9	22	Multi-layered optimizing migration
10	23	Reverse-engineered context-sensitive hardwa...

Total Quantity Ordered per Customer

Insight

This query uses a Common Table Expression (CTE) to calculate how many books each customer has ordered, based on the sum of quantity grouped by customer in the orders table.

Query

```sql
-- CTE: Customers and number of books ordered
WITH customer_book_counts AS (
    SELECT
        c.customer_id,
        c.name,
        SUM(o.quantity) AS total_books_ordered
    FROM
        customers c
    JOIN
        orders o ON c.customer_id = o.customer_id
    GROUP BY
        c.customer_id,
        c.name
)
SELECT
    *
FROM
    customer_book_counts
ORDER BY
    customer_id;
```

Query Output

	customer_id [PK] integer	name character varying (100)	total_books_ordered bigint
1	2	Crystal Clements	10
2	6	Stephen Vasquez	4
3	7	Susan Hicks	1
4	8	Matthew Johnson	8
5	10	Ronald Osborn	6
6	11	Thomas Garcia	5
7	12	Jennifer Murray	3
8	13	Kristine Kim	14
9	14	John Wood	12
10	15	Vanessa Gaines	10

Top Spenders: Customer Ranking

Insight

This query ranks customers based on the total amount they have spent using a window function. It shows who contributed most to revenue and helps prioritize high-value users.

Query

```sql
-- Rank customers based on total amount spent
SELECT
    customer_id,
    total_amount_sum,
    RANK() OVER (
        ORDER BY total_amount_sum DESC
    ) AS spending_rank
FROM (
    SELECT
        customer_id,
        SUM(total_amount) AS total_amount_sum
    FROM
        orders
    GROUP BY
        customer_id
) AS customer_totals;
```

Query Output

	customer_id integer	total_amount_sum numeric	spending_rank bigint
1	457	1398.90	1
2	174	1080.95	2
3	364	1052.27	3
4	405	991.00	4
5	386	986.30	5
6	425	942.62	6
7	474	929.19	7
8	163	746.65	8
9	167	719.93	9
10	214	682.15	10

Most Expensive Book In Each Genre

Insight

This query returns the highest-priced book in every genre by combining filtering with grouping. It helps understand price distribution and identify premium books across different categories.

Query

```sql
-- Highest priced book in each genre
SELECT
    title,
    genre,
    price
FROM
    books
WHERE
    (genre, price) IN (
        SELECT
            genre,
            MAX(price)
        FROM
            books
        GROUP BY
            genre
    );
```

Query Output

	title character varying (100)	genre character varying (50)	price numeric (10,2)
1	Function-based heuristic analyzer	Biography	49.53
2	Synchronized client-server service-desk	Fiction	49.89
3	Switchable modular moratorium	Romance	49.88
4	Optimized content-based standardization	Science Fiction	49.96
5	Stand-alone content-based hub	Fantasy	49.90
6	Proactive system-worthy orchestration	Mystery	49.98
7	Profound tertiary encoding	Non-Fiction	49.34

Conclusion

- Through these queries, I understood how to extract real insights from data.
- I learned how to find top customers, best-selling books, and order trends.
- Writing queries improved my grip on joins, grouping, and filters.
- This project showed how SQL can be used to solve practical, real-world business problems.

Thank you for Viewing My SQL Project.

By:
Abdul Razzaq