Bookstore Data Analysis Using PostgreSQL

April 2025

Submitted By:
Vamshi
Aspiring Data Analyst

This project showcases real-world SQL applications by performing data analysis on a bookstore database. Key tasks include data extraction, transformation, and insightful query writing using PostgreSQL.

```
--DROP TABLE IF THE TABLES IS EXISTS

DROP TABLE IF EXISTS Books;

DROP TABLE IF EXISTS Customers;

DROP TABLE IF EXISTS Orders;



CREATE TABLE Books(

Book_ID SERIAL PRIMARY KEY,

Title VARCHAR(100),

Author VARCHAR(100),

Genre VARCHAR(100),

Published_Year INT,

Price NUMERIC(10,2),

Stock INT

);


--CREATE TABLE CUSTOMERS


DROP TABLE IF EXISTS Customers;


CREATE TABLE Customers(

Customer_ID SERIAL PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100),

Phone VARCHAR(100),

City VARCHAR(100),

Country VARCHAR(100)

);
```

```
--CREATE TABLE ORDERS

DROP TABLE IF EXISTS Orders;

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)
);

--TABLES
SELECT * FROM BOOKS;
SELECT * FROM CUSTOMERS;
SELECT * FROM ORDERS;

--INSERTING THE DATA INTO THE TABLE BOOKS
COPY BOOKS(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)
FROM 'C:\Users\vamshi\Downloads\Books.csv'
DELIMITER ','
CSV HEADER;
```

```
--INSERTING THE DATA INTO THE TABLE CUSTOMERS

COPY CUSTOMERS(Customer_ID, Name, Email, Phone, City, Country)
FROM 'C:\Users\vamshi\Downloads\Orders.csv'
DELIMITER ','
CSV HEADER;

--INSERTING THE DATA INTO THE TABLE ORDERS

COPY ORDERS(Order_ID, Customer_ID, Book_ID, Order_Date, Quantity, Total_Amount)
FROM 'C:\Users\vamshi\Downloads\Orders.csv'
DELIMITER ','
CSV HEADER;

--1)RETRIVE ALL BOOKS IN FICTION GENER

SELECT * FROM BOOKS
WHERE Genre = 'Fiction';

--2)FIND THE BOOK PUBLISHED AFTER THE YEAR 1950

SELECT * FROM BOOKS
WHERE PUBLISHED_YEAR > 1950;
```

```
--3)LIST ALL THE CUSTOMERS FROM CANADA

SELECT * FROM CUSTOMERS
WHERE COUNTRY = 'Canada';

--4)SHOW ORDER PLACED IN NOVEMBER 2023

SELECT * FROM ORDERS
WHERE ORDER_DATE BETWEEN '2023-11-01' AND '2023-11-30';

--5)RETRIVE THE TOTAL STOCK OF BOOKS AVALIABLE

SELECT SUM(Stock)
FROM BOOKS;

--6)FIND THE DETAILS OF MOST EXPENSIVE BOOK

SELECT * FROM BOOKS
ORDER BY PRICE DESC
LIMIT 1;

--7)SHOW ALL CUSTOMERS ORDERED MORE THAN 1 QUANTITY OF A BOOK

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

```
--8)RETRIVE ALL THE ORDERS WHERE THE TOTAI_AMOUNT EXCEEDS $20

SELECT * FROM ORDERS
WHERE TOTAL_AMOUNT > 20;



--9)LIST ALL GENRES AVALIABLE IN THE BOOKS TABLE
SELECT DISTINCT (GENRE)
FROM BOOKS;

--10)FIND THE BOOK WITH THE LOWEST STOCK

SELECT * FROM BOOKS
ORDER BY STOCK ASC
LIMIT 1;

--11)CALCULATE THE TOTAL REVENUE GENERATED FROM ALL ORDERS
SELECT
SUM(TOTAL_AMOUNT)AS REVENUE_GENERATED
FROM ORDERS;



--11)RETRIVE THE TOTAL NUMBER OF BOOKS SOLD FOR EACH GENRE
SELECT b.GENRE,SUM(o.QUANTITY) AS TOTAI_BOOKS_SOLD
FROM ORDERS o
join books b
on o.book_id=b.book_id
group by b.GENRE;
```

```
--12)FIND THE AVERAGE PRICE OF THE BOOKS IN THE "FANTASY" GENRE

SELECT AVG(PRICE) AS AVERAGE_PRICE
FROM BOOKS
WHERE GENRE='Fantasy';

--13)LIST CUSTOMERS WHO HAVE PLACED AT LEAST TWO ORDERS

SELECT customer_id,COUNT(ORDER_ID) AS ORDER_COUNT
FROM ORDERS
GROUP BY CUSTOMER_ID
HAVING COUNT(ORDER_ID)>=2;

--BY USING JOIN

SELECT o.Customer_id,c.name,COUNT(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;
```

```
--14)FIND THE MOST FREQUENTLY ORDERD BOOK

SELECT BOOK_ID,COUNT(ORDER_ID)AS ORDER_COUNT
FROM ORDERS
GROUP BY BOOK_ID
ORDER BY ORDER_COUNT DESC
LIMIT 1;

--BY USING JOIN

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;
```

```sql
--15)SHOW THE TOP 3 MOST EXPENSIVE BOOKS BY 'FANTASY' GENRE

SELECT * FROM BOOKS
WHERE GENRE='Fantasy'
ORDER BY PRICE DESC
LIMIT 3;

--16)RETRIVE 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;

--17)LIST THE CITIES WHERE CUSTOMERS WHO SPENT OVER $30 ARE LOCATED

SELECT DISTINCT c.CITY,o.TOTAL_AMOUNT AS SPENT
FROM ORDERS o
JOIN CUSTOMERS c
ON o.customer_id = c.customer_id
where o.total_amount > 30;
```

```
--18)FIND THE CUSTOMERS 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;


--19)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;
```