Online-Book-Store-

"Developed an **Online Book Store SQL Project** to manage and streamline book inventory, customer orders, and sales data. Designed and implemented a relational database schema, wrote complex SQL queries, and optimized database performance to ensure seamless data retrieval and efficient operations.

Online Book Store SQL Project

This project is a database management system for an **Online Book Store**, designed to efficiently manage book inventory, customer orders, and sales data. It includes a relational database schema, SQL queries, and optimized database operations.

Features

- **Database Schema**: Designed to store book details, customer information, orders, and sales data.
- **SQL Queries**: Includes complex queries for retrieving book details, generating sales reports, and managing inventory.
- **Optimized Performance**: Ensures fast and efficient data retrieval and storage.

Technologies Used

- **Database**: MySQL
- **Tools**: MySQL Workbench, Git, GitHub

Database Schema

The database consists of the following tables:

- `Books`: Stores book details (title, author, price, stock).
- `Customers`: Stores customer information (name, email, address).
- `Orders`: Manages customer orders (order ID, customer ID, book ID, quantity, order date).
- `Sales`: Tracks sales data (sale ID, book ID, customer ID, sale date, total amount).

SQL Queries Examples

1. Retrieve all books in stock:

```
SELECT * FROM Books WHERE stock > 0;```
```

CREATE DATABASE

```
CREATE DATABASE ONLINEBOOKSTORE;
```

USE THAT DATABASE

```
USE ONLINEBOOKSSTORE;
```

CREATE TABLE WITH THE NAME OF BOOKS

```
CREATE TABLE BOOKS (
BOOK_ID INT PRIMARY KEY,
Title VARCHAR(100),
Author VARCHAR(100),
Genre VARCHAR(150),
Published_Year INT,
Price FLOAT,
Stock INT
);
```

SEEN ALL THE RECORDS IN BOOKS TABLE

-- THIS IS FIRST TABLE

```
SELECT * FROM BOOKS;
```

CREATE SECOND CUSTOMERS TABLE

```
CREATE TABLE CUSTOMERS(
Customer_ID INT PRIMARY KEY,
Name VARCHAR(200),
Email VARCHAR(400),
Phone BIGINT,
City VARCHAR(500),
Country VARCHAR(1000)
);
```

SEEN ALL THE RECORDS IN CUSTOMERS TABLE

```
SELECT * FROM CUSTOMERS;
```

CREATE THIRD TABLE

```
CREATE TABLE ORDERS(
Order_ID INT NOT NULL,
Customer_ID     INT,
Book_ID INT,
Order_Date      TEXT,
Quantity INT,
Total_Amount FLOAT
);
```

SEEN ALL THE RECORDS IN ORDERS TABKE

```
SELECT * FROM ORDERS;
```

Import Data into Books Table

```
COPY Books(Book_ID, Title, Author, Genre, Published_Year, Price, Stock)
FROM 'D:\Course Updates\30 Day Series\SQL\CSV\Books.csv'
CSV HEADER;
```

Import Data into Customers Table

```
COPY Customers(Customer_ID, Name, Email, Phone, City, Country)
FROM 'D:\Course Updates\30 Day Series\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:\Course Updates\30 Day Series\SQL\CSV\Orders.csv'
CSV HEADER;
```

1) Retrieve all books in the "Fiction" genre:

```
SELECT * FROM BOOKS
WHERE GENRE = 'FICTION';
```

2) Find books published after the year 1950:

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

3) List all customers from the Canada:

```
SELECT * FROM CUSTOMERS
WHERE CITY= 'CANADA';
```

4) Show orders placed in November 2023:

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

5) Retrieve the total stock of books available:

```
SELECT SUM(STOCK) FROM BOOKS;
```

6) Find the details of the most expensive book:

```
SELECT * FROM BOOKS
ORDER BY PRICE DESC
LIMIT 1;
```

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

```
SELECT * FROM ORDERS
WHERE QUANTITY > 1
ORDER BY QUANTITY;
```

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

```
SELECT * FROM ORDERS
WHERE TOTAL_AMOUNT > 20
ORDER BY TOTAL_AMOUNT;
```

9) List all genres available 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 20;
```

11) Calculate the total revenue generated from all orders:

```
SELECT ROUND(SUM(TOTAL_AMOUNT),2) FROM ORDERS;
```

ADVANCED QUESTIONS :

1) Retrieve the total number of books sold for each genre:

```
SELECT GENRE,SUM(BOOK_ID) AS TOTAL_COUNT FROM BOOKS
GROUP BY GENRE
ORDER BY TOTAL_COUNT DESC;
```

2) Find the average price of books in the "Fantasy" genre:

```
SELECT GENRE ,ROUND(AVG(PRICE),2) FROM BOOKS
GROUP BY GENRE
HAVING GENRE ='FANTASY';
```

3) 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
ORDER BY COUNT(Order_id) DESC;
```

4) 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;
```

5) Show the top 3 most expensive books of 'Fantasy' Genre :

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

6) Retrieve the total quantity of books sold by each author:

```
SELECT AUTHOR, SUM(QUANTITY) AS TOTAL_QTY FROM BOOKS
JOIN ORDERS
ON BOOKS.BOOK_ID = ORDERS.BOOK_ID
GROUP BY AUTHOR
ORDER BY TOTAL_QTY DESC;
```

7) List the cities where customers who spent over $30 are located:

```
SELECT CITY,TOTAL_AMOUNT FROM CUSTOMERS
JOIN ORDERS
ON CUSTOMERS.CUSTOMER_ID = ORDERS.CUSTOMER_ID
WHERE TOTAL_AMOUNT > 30
GROUP BY CITY,TOTAL_AMOUNT
ORDER BY TOTAL_AMOUNT DESC;
```

8) 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;
```

9) Calculate the stock remaining after fulfilling all orders:

```
SELECT b.book_id, b.title, b.stock,SUM(o.quantity) AS Order_quantity,
        b.stock- SUM(o.quantity) 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;
```