ONLINE BOOK STORE DATABASE PROJECT – MYSQL

This project demonstrates the design and implementation of an Online Book Store database using MySQL. It includes database creation, table design, relationships, and a collection of basic and advanced SQL queries to perform realworld operations such as retrieving data, calculating revenue, tracking orders, and more.

Technologies Used

- MySQL
- SQL Queries
- Database Design
- Joins and Aggregation

Database Schema

The database contains the following core tables:
- Books
- Customer
- Orders

Primary keys and foreign keys are used to maintain referential integrity between tables.

Database and Tables Creation

```
CREATE DATABASE OnlineBookStore;
USE OnlineBookStore;

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 Customer(
Customer_ID SERIAL PRIMARY KEY,
Name VARCHAR(100),
Email VARCHAR(100),
Phone VARCHAR(100),
City VARCHAR(50),
Country VARCHAR(150)
);
```

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

BASIC SQL QUESTIONS WITH ANSWERS

Question: Retrieve all Books in the 'Fiction' genre

SQL Query:

```
SELECT * FROM Books WHERE Genre = 'Fiction';
```

Question: Find Books published after the year 1950

SQL Query:

```
SELECT * FROM Books WHERE Published_Year > 1950;
```

Question: List all the customers from Canada

SQL Query:

```
SELECT * FROM Customer WHERE Country = 'Canada';
```

Question: Show orders placed in November 2023

SQL Query:

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

Question: Retrieve the total stock of books available

SQL Query:

```
SELECT SUM(stock) AS 'Total Stock of Books Available' FROM Books;
```

Question: Find the details of the most expensive book

SQL Query:

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

Question: Show all customers who ordered more than 1 quantity of books

SQL Query:

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

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

SQL Query:

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

Question: List all distinct genres available in the Books table

SQL Query:

```
SELECT DISTINCT Genre FROM Books;
```

Question: Find the book(s) with the lowest stock

SQL Query:

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

Question: Calculate the total revenue from all order

SQL Query:

```
SELECT SUM(Total_Amount) AS 'Revenue' FROM Orders;
```

ADVANCED SQL QUESTIONS WITH ANSWERS

Question: Retrieve the total number of books sold for each genre

SQL Query:

```
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;
```

Question: Find the average price of books in the 'Fantasy' genre

SQL Query:

```
SELECT Genre, AVG(Price) AS Average_Price
FROM Books WHERE Genre = 'Fantasy'
GROUP BY Genre;
```

Question: List customers who have placed at least two orders

SQL Query:

```
SELECT c.Name, o.Customer_ID, COUNT(*) AS Number_Of_Ordered_Books

FROM Orders o

JOIN Customer c ON o.Customer_ID = c.Customer_ID

GROUP BY c.Name, o.Customer_ID HAVING COUNT(*) >= 2;
```

Question: Find the most frequently ordered book

SQL Query:

```
SELECT o.Book_ID, b.Title, COUNT(*) 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;
```

Question: Show the top 3 most expensive books of the 'Fantasy' genre

SQL Query:

```
SELECT * FROM Books
WHERE Genre ='Fantasy'
ORDER BY Price DESC LIMIT 3;
```

Question: Retrieve the total quantity of books sold by each author

SQL Query:

```
SELECT b.Author, SUM(o.Quantity) AS Total_Sold_Books

FROM  Orders  o JOIN Books b ON o.Book_ID = b.Book_ID GROUP BY b.Author;
```

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

SQL Query:

```
SELECT DISTINCT c.City, Total_Amount

FROM Orders o
JOIN Customer c ON o.Customer_ID = c.Customer_ID
WHERE o.Total_Amount > 30;
```

Question: Find the customer who spent the most on orders

SQL Query:

```
SELECT c.Customer_ID, c.Name, SUM(o.Total_Amount) AS Total_Spent
FROM Orders o
JOIN Customer c ON o.Customer_ID = c.Customer_ID
GROUP BY c.Customer_ID, c.Name
ORDER BY Total_Spent DESC LIMIT 1;
```

Question: Calculate the stock remaining after fulfilling all orders

SQL Query:

```
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;
```