PROJECT TITLE: ONLINE BOOK STORE

Introduction:

The Online Book Store is a structured SQL-based project that simulates a real-world online bookstore system using relational databases. The project aims to efficiently manage and analyse data related to books, customers, and their purchases. The database created for this project is named online_bookstore_project, and it contains three interrelated tables designed to represent a simplified version of an online book retail platform.

Database Structure:

1. Books Table:

The Books table stores information about each book available in the store. It includes the following columns:

- Book_ID (Primary Key): It Shows the Book ID Number.
- Title: Name of the Book.
- Author: Writer of the Book.
- Genre: a particular type or style of literature that you can recognize because of its special characteristics.
- Published_Year: The specific year when a book was officially released or made accessible to the public.
- Price: Cost of the Book in Dollars ($).
- Stock: Number of units of books available in stock.

2. Customers Table:

The Customers table maintains information about all the registered customers. Its columns include:

- Customer_ID (Primary Key): A unique identification assigned to a customer, used to track and manage their information, interactions, and transactions.
- Name: Name of the customers
- Email: Customer Email ID for inquiries, support, feedback, or other communication.
- Phone: Customer Phone Number for Communication.
- City: Customer City Name for market analysis, sales and marketing strategies, customer segmentation, and delivery optimization.
- Country: Customer Country Name for personalize the shopping experience, determine applicable taxes and regulations, and manage international shipping and logistics.

3. Orders Table:

The Orders table records all transactions made by customers. It captures key order details with the following columns:

- Order_ID (Primary Key): It helps to track, manage, and verify orders throughout the fulfilment process. It also Shows number of orders placed by customers.
- Customer_ID (Foreign Key referencing Customers.Customer_ID)
- Book_ID (Foreign Key referencing Books.Book_ID)
- Order_Date: Order_Date is used to record the date and time when a customer placed an order of books.
- Quantity: Quantity shows the number of copies of a specific book that have been sold.
- Total_Amount: Total_Amount Paid By the Customer On Their Orders.

Project Objectives:

This project focuses on querying and analysing the database to extract meaningful insights using both basic and advanced SQL queries. Key objectives include:

- Fetching customer details based on order patterns
- Calculating total spending and identifying high-value customers
- Determining best-selling books
- Managing inventory by tracking stock remaining after sales
- Performing data-driven decision-making using grouped and aggregated data

By combining real-world business logic with SQL operations like JOINs, GROUP BY, HAVING, aggregation functions, and subqueries, this project showcases the practical application of relational database concepts in e-commerce platforms.

CREATEING DATABASE: -

```
CREATE DATABASE ONLINE_BOOKSTORE_PROJECT;
```

 # STEP 1. CREATE FIRST TABLE "BOOKS": -

```
CREATE TABLE BOOKS (
    Book_ID  INT   PRIMARY KEY,
    Title        VARCHAR (100),
    Author     VARCHAR (100),
    Genre       VARCHAR (100),
```

Published_Year INT,

Price FLOAT,

Stock INT);

 # STEP 2. CHECK THE CREATED TABLE BOOKS: -

- SELECT * FROM BOOKS;
- DESC BOOKS;

 # STEP 3. CREATE SECOND TABLE "CUSTOMERS": -

- CREATE TABLE CUSTOMERS (

  Customer_ID INT PRIMARY KEY,

  Name VARCHAR (100),

  Email VARCHAR (100),

  Phone INT,

  City VARCHAR (100),

  Country VARCHAR (100)) ;

 # STEP 4. CHECK THE CREATED TABLE CUSTOMERS: -

- SELECT * FROM CUSTOMERS;
- DESC CUSTOMERS;

 # STEP 5. CREATE THE THIRD TABLE "ORDERS": -

- CREATE TABLE ORDERS (

  Order_ID INT PRIMARY KEY,

  Customer_ID INT references customers (customer_id),

  Book_ID INT references books (Book_ID),

  Order_Date DATE,

  Quantity INT,

  Total_Amount FLOAT) ;

 # STEP 6. IMPORT DATA DIRECTLY FROM THE CSV FILE INTO CREATED TABLES :-

1. GO TO DATABASE

2. OPEN THE TABLES CREATED

3. RIGHT CLICK ON THE REQUIRED TABLE

4. SELECT OPTION TABLE DATA IMPORT WIZARD

5. CLICK ON NEXT OPTIONS

6. FINISH THE IMPORT PROCESS

 # STEP 7. CHECK ALL THE TABLES AFTER DATA IMPORT: -

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

BASIC QUERIES QUESTIONS:

1. RETRIVE ALL BOOKS IN THE "FICTION" GENRE: -

- SELECT BOOK_ID, TITLE FROM BOOKS WHERE GENRE = "FICTION";

2. FIND BOOKS PUBLISHED AFTER THE YEAR 1950: -

- SELECT BOOK_ID, TITLE FROM BOOKS WHERE PUBLISHED_YEAR>1950;

3. LIST ALL THE CUSTOMERS FROM CANADA: -

- SELECT * FROM CUSTOMERS WHERE COUNTRY = "CANADA";

4. SHOW ALL ORDERS PLACED IN NOVEMBER 2023: -

- SELECT ORDER_ID, CUSTOMER_ID, BOOK_ID FROM orders WHERE ORDER_DATE BETWEEN "2023-11-01" AND "2023-11-30";

5. RETRIVE THE TOTAL STOCKS OF BOOKS AVAILABLE: -

- SELECT SUM(STOCK) AS TOTAL_BOOKS_AVAILABLE_IN_STOCKS FROM BOOKS;

6. FIND THE DETAILS OF MOST EXPENSIVE BOOKS: -

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

7. SHOW ALL CUSTOMERS WHO ORDER MORE THAN 5 QUANTITIES OF BOOKS: -

- SELECT C.CUSTOMER_ID, C.NAME, O.QUANTITY
  FROM ORDERS O
  JOIN CUSTOMERS C
  ON C.CUSTOMER_ID = O.CUSTOMER_ID
  WHERE O.QUANTINTY > 5
  ORDER BY O.QUANTITY DESC;

8. RETRIVE ALL ORDERS WHERE THE AMOUNT EXCEED $50: -

- SELECT * FROM ORDERS WHERE TOTAL_AMOUNT > 50.0;

9. LIST ALL THE GENRES AVAILABLE IN BOOKS TABLE: -

- SELECT DISTINCT GENRE FROM BOOKS;

10. FIND THE BOOK WITH THE LOWEST STOCK: -

- SELECT * FROM BOOKS ORDER BY STOCK LIMIT 1;

11. CALCULATE THE TOTAL REVENUE GENERATED FROM ROMANCE GENRE: -

- SELECT SUM(PRICE) AS TOTAL_REVENUE_BY_GENRE_ROMANCE
  FROM BOOKS WHERE GENRE = "ROMANCE”;

ADVANCED QUERIES QUESTIONS:

12. RETRIVE THE TOTAL NUMBER OF BOOKS SOLD FOR EACH GENURE: -

- 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;

13. FIND THE AVERAGE PRICE OF BOOKS IN THE FANTASY GENRE: -

- SELECT AVG(PRICE) AS AVERAGE_PRICE FROM BOOKS WHERE GENRE = "FANTASY";

14. LIST THE CUSTOMERS WHO HAVE PLACED ATLEAST 2 ORDERS: -

- SELECT C.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 C.CUSTOMER_ID, C.NAME

  HAVING COUNT(O.ORDER_ID) > = 2;

15. FIND THE MOST FREQUENT BOOK ORDER: -

- SELECT B.BOOK_ID, B.TITLE, COUNT (O.ORDER_ID) AS MOST_FREQUENTLY_BOOK_ORDER

  FROM ORDERS O

  JOIN BOOKS B

  ON O.BOOK_ID = B.BOOK_ID

  GROUP BY B.BOOK_ID, B.TITLE

  ORDER BY COUNT(O.ORDER_ID) DESC LIMIT 1;

16. SHOW THE TOP 3 MOST EXPENSIVE BOOKS OF FANTASY GENRE: -

- SELECT * FROM BOOKS WHERE GENRE = "FANTASY" ORDER BY PRICE DESC LIMIT 3 ;

17. RETRIVE THE TOTAL QUANTITY OF BOOKS SOLD BY EACH AUTHOR: -

- SELECT B.AUTHOR, SUM (O.QUANTITY) AS TOTAL_QUANTITY_OF_BOOKS_SOLD

  FROM ORDERS O

  JOIN BOOKS B

  ON O.BOOK_ID = B.BOOK_ID

  GROUP BY B.AUTHOR;

18. LIST THE CITIES WHERE CUSTOMERS WHO SPEND OVER $30 ARE LOCATED: -

- SELECT DISTINCT C.CITY, O.TOTAL_AMOUNT AS TOTAL_AMOUNT_SPEND
  FROM ORDERS O
  JOIN CUSTOMERS C
  ON O.CUSTOMER_ID = C.CUSTOMER_ID
  WHERE O.TOTAL_AMOUNT > 30 ;

19. FIND THE CUSTOMER WHO SPEND THE MOST ON ORDERS: -

- SELECT C.CUSTOMER_ID, C.NAME, SUM (O.TOTAL_AMOUNT) AS TOTAL_SPEND
  FROM ORDERS O
  JOIN CUSTOMERS C
  ON O.CUSTOMER_ID = O.CUSTOMER_ID
  GROUP BY C.CUSTOMER_ID, C.NAME
  ORDER BY TOTAL_SPEND DESC LIMIT 1;

20. CALCULATE THE STOCK REMAINING AFTER FULLFILLING ALL ORDERS: -

- SELECT B.BOOK_ID, B.TITLE,
  B.STOCK - COALESCE(SUM(O.QUANTITY), 0) AS REMAINING_STOCK
  FROM BOOKS B
  LEFT JOIN ORDERS O
  ON B.BOOK_ID = O.BOOK_ID
  GROUP BY B.BOOK_ID, B.TITLE, B.STOCK ;

THE END