**Project Submission: Online Retail Application Database**

**Abstract**

With the growth of e-commerce, online retail application databases have become popular SQL project ideas. These applications allow customers to register and purchase items online. This project involves creating a database to manage customer registrations, product inventories, and order transactions. It demonstrates the interconnection between customers, products, and orders, providing hands-on experience in designing a relational database for a real-world scenario.

**Introduction**

E-commerce platforms enable customers to register and purchase products online. The registration process generates a unique customer ID and collects essential information such as name, address, contact details, and bank information. Upon purchasing a product, a bill is generated based on the quantity, price, and any applicable discounts. The customer selects a payment method to complete the transaction, and the product is delivered to the chosen location.

**Objectives**

* Design a relational database to manage customer registrations, product inventories, and order transactions.
* Establish relationships between customers, products, and orders.
* Implement SQL queries to perform typical database operations.

**Database Design**

**Entity-Relationship Diagram (ERD)**

[Insert ERD diagram here]

**Database Schema**

The database schema includes the following tables:

* **Customers**: Contains information about customers.
* **Products**: Contains information about products available for purchase.
* **Orders**: Contains information about customer orders.
* **OrderDetails**: Contains information about the details of each order.

**SQL Scripts**

**1. Create the Database**

CREATE DATABASE ECommerceDB;

USE ECommerceDB;

**2. Create Tables**

**Customers Table**

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerName VARCHAR(100) NOT NULL,

Address VARCHAR(255) NOT NULL,

ContactInfo VARCHAR(100) NOT NULL,

Email VARCHAR(100) NOT NULL UNIQUE,

Password VARCHAR(100) NOT NULL,

BankDetails VARCHAR(255) NOT NULL

);

**Products Table**

CREATE TABLE Products (

ProductID INT PRIMARY KEY AUTO\_INCREMENT,

ProductName VARCHAR(100) NOT NULL,

Price DECIMAL(10, 2) NOT NULL,

Stock INT NOT NULL

);

**Orders Table**

CREATE TABLE Orders (

OrderID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

OrderDate DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,

TotalAmount DECIMAL(10, 2) NOT NULL,

DeliveryAddress VARCHAR(255) NOT NULL,

PaymentMethod VARCHAR(50) NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**OrderDetails Table**

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

ProductID INT,

Quantity INT NOT NULL,

Price DECIMAL(10, 2) NOT NULL,

Discount DECIMAL(10, 2) DEFAULT 0,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

**3. Insert Sample Data**

**Insert Customers**

INSERT INTO Customers (CustomerName, Address, ContactInfo, Email, Password, BankDetails) VALUES('charan', '123 Elm St, Springfield', '123-456-7890', 'charan.sr@gmail.com', 'password123', 'Bank XYZ, Account 1234567890'),('priya', '456 Oak St, Springfield', '987-654-3210', 'priya.sr@gmail.com', 'password456', 'Bank ABC, Account 0987654321');

**Insert Products**

INSERT INTO Products (ProductName, Price, Stock) VALUES

('Laptop', 1000.00, 50),

('Smartphone', 700.00, 100),

('Headphones', 50.00, 200),

('Keyboard', 30.00, 150);

**Insert Orders**

INSERT INTO Orders (CustomerID, TotalAmount, DeliveryAddress, PaymentMethod) VALUES

(1, 1050.00, '123 Elm St, Springfield', 'Credit Card'),

(2, 730.00, '456 Oak St, Springfield', 'Debit Card');

**Insert OrderDetails**

INSERT INTO OrderDetails (OrderID, ProductID, Quantity, Price, Discount) VALUES

(1, 1, 1, 1000.00, 0),

(1, 3, 1, 50.00, 0),

(2, 2, 1, 700.00, 0),

(2, 4, 1, 30.00, 0);

**Query Examples**

**List All Customers**

SELECT \* FROM Customers;

**List All Products**

SELECT \* FROM Products;

**List All Orders**

SELECT \* FROM Orders;

**List All Order Details**

SELECT \* FROM OrderDetails;

**List All Orders by a Specific Customer**

SELECT \* FROM Orders

WHERE CustomerID = 1; -- Replace with the desired CustomerID

**List Order Details for a Specific Order**

SELECT od.\*, p.ProductName

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

WHERE od.OrderID = 1; -- Replace with the desired OrderID

**Calculate Total Sales for Each Product**

SELECT p.ProductName, SUM(od.Quantity \* od.Price) AS TotalSales

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName;

**List All Products with Stock Less Than a Specified Amount**

SELECT \* FROM Products

WHERE Stock < 50; -- Replace with the desired stock amount

**List All Customers Who Have Placed Orders**

SELECT DISTINCT c.CustomerName, c.Email

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID;

**List Total Amount Spent by Each Customer**

SELECT c.CustomerName, c.Email, SUM(o.TotalAmount) AS TotalSpent

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerName, c.Email;

**Advanced Queries**

**List the Most Popular Products (Based on Quantity Sold)**

SELECT p.ProductName, SUM(od.Quantity) AS TotalQuantitySold

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName

ORDER BY TotalQuantitySold DESC;

**List All Orders Along with Customer and Order Details**

SELECT o.OrderID, o.OrderDate, o.TotalAmount, c.CustomerName, c.Email, p.ProductName, od.Quantity, od.Price, od.Discount

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID;

**List Revenue Generated Each Day**

SELECT DATE(OrderDate) AS OrderDay, SUM(TotalAmount) AS DailyRevenue

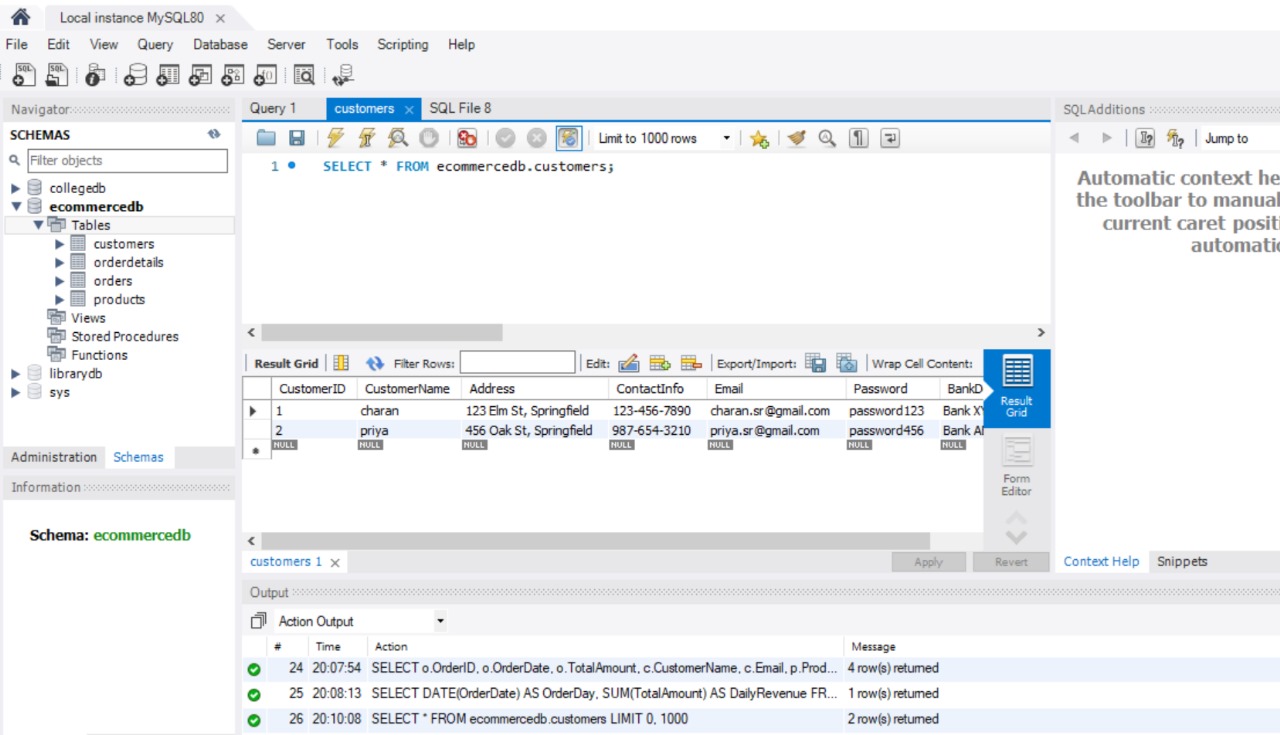
FROM Orders

GROUP BY OrderDay;

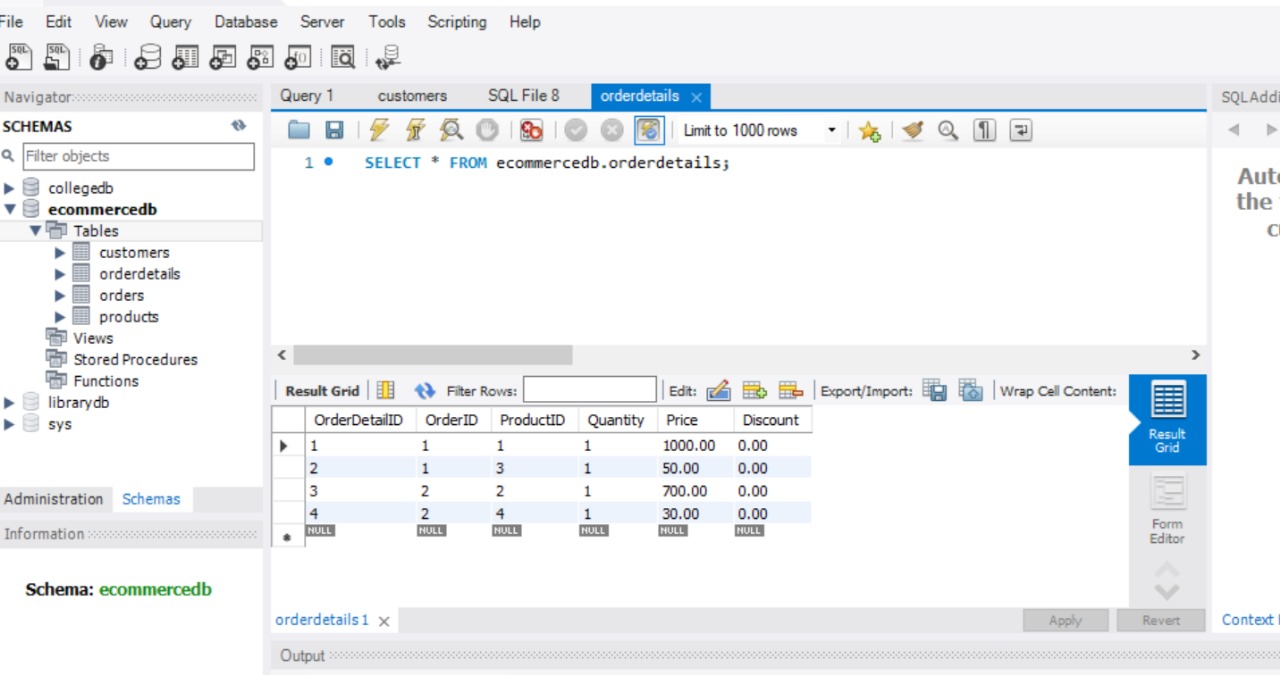
**Conclusion**

This project demonstrates the creation and management of a relational database for an online retail application. It includes designing the database schema, inserting sample data, and performing various queries to manage and retrieve information about customers, products, and orders. This system supports essential e-commerce functionalities and provides a solid foundation for further development and enhancements.

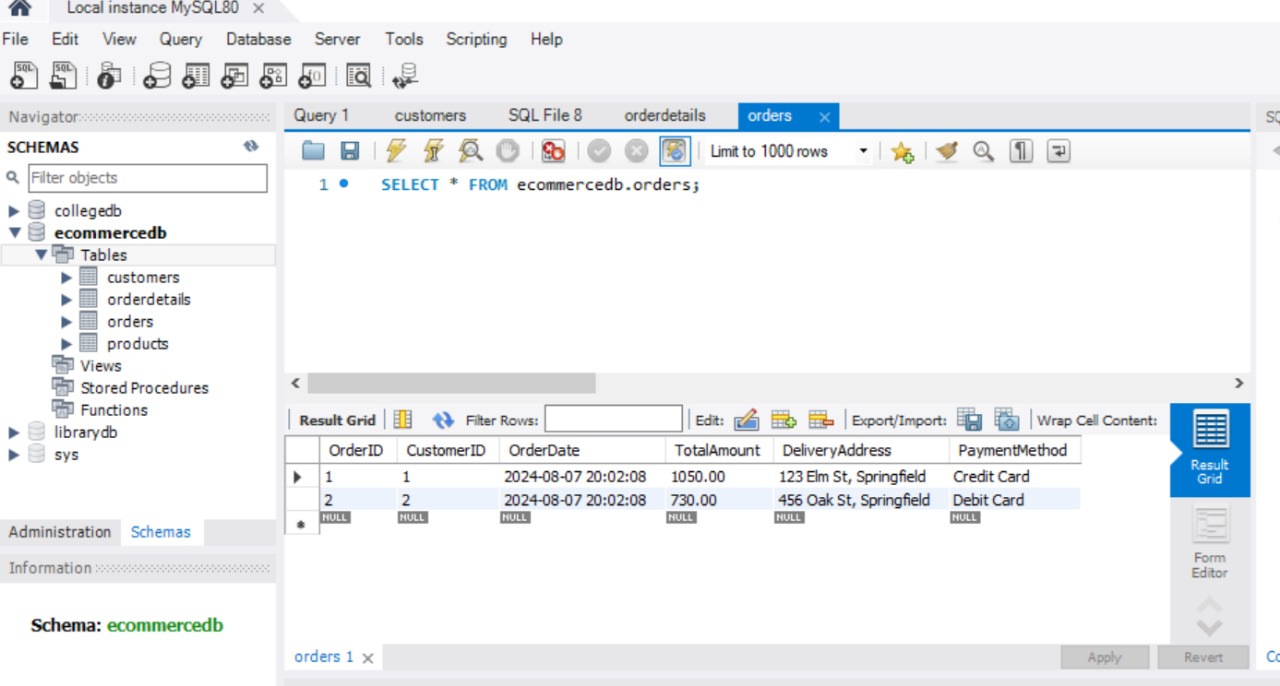
**Output**



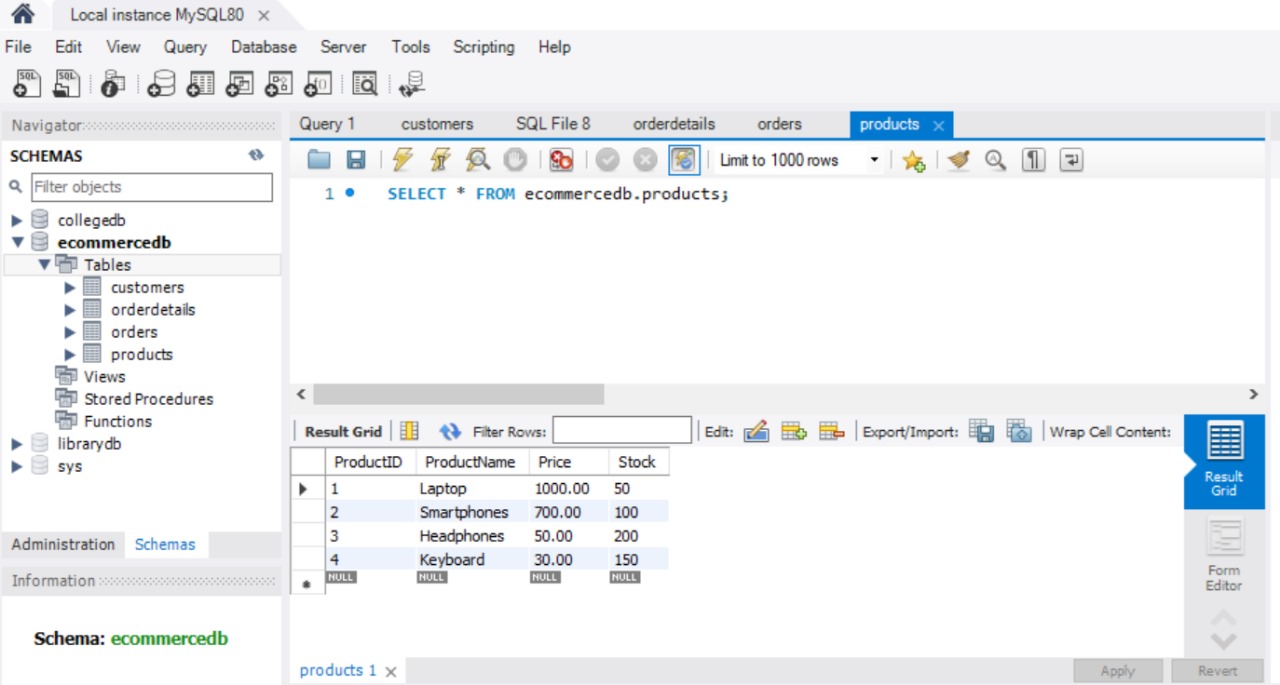
**customers**



**Order details**



**Orders**



**products**