E-Commerce Database Design

# Overview

This document outlines the database design for an e-commerce platform, covering product catalog management, customer information, order processing, payment handling, and supplier management. The goal is to design an efficient relational database structure that supports key features of an online store.

# Database Schema

The schema is divided into several core tables that represent essential aspects of the e-commerce system:

1. Products Table: Stores product information including name, description, price, and stock quantity.  
2. ProductCategories Table: Stores product categories such as Electronics, Clothing, etc.  
3. Customers Table: Stores customer information like name, contact details, and shipping address.  
4. Orders Table: Stores details of customer orders, including order status and total amount.  
5. OrderDetails Table: Stores individual items within an order, linking products to orders.  
6. ShoppingCart Table: Manages products added to the shopping cart before the customer completes a purchase.  
7. Payments Table: Stores payment information for completed orders.  
8. Suppliers Table: Stores supplier information who provide products for the e-commerce platform.

# 1. Table Definitions

## 1.1 Suppliers Table

The Suppliers table stores details of suppliers who provide products for the online store.

SQL Code:  
CREATE TABLE Suppliers (  
 SupplierID INT PRIMARY KEY,  
 SupplierName VARCHAR(255) NOT NULL,  
 ContactName VARCHAR(255),  
 ContactPhone VARCHAR(15),  
 ContactEmail VARCHAR(255)  
);

## 1.2 ProductCategories Table

The ProductCategories table stores categories for the products.

SQL Code:  
CREATE TABLE ProductCategories (  
 CategoryID INT PRIMARY KEY,  
 CategoryName VARCHAR(255) NOT NULL  
);

## 1.3 Products Table

The Products table stores details of products, including their price, stock quantity, and foreign key references to categories and suppliers.

SQL Code:  
CREATE TABLE Products (  
 ProductID INT PRIMARY KEY,  
 ProductName VARCHAR(255) NOT NULL,  
 Description TEXT,  
 Price DECIMAL(10, 2) NOT NULL,  
 StockQuantity INT NOT NULL,  
 CategoryID INT,  
 SupplierID INT,  
 CONSTRAINT FK\_Category FOREIGN KEY (CategoryID) REFERENCES ProductCategories(CategoryID),  
 CONSTRAINT FK\_Supplier FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)  
);

## 1.4 Customers Table

The Customers table stores information about customers, such as their contact details and shipping address.

SQL Code:  
CREATE TABLE Customers (  
 CustomerID INT PRIMARY KEY,  
 FirstName VARCHAR(100),  
 LastName VARCHAR(100),  
 Email VARCHAR(255) UNIQUE,  
 PhoneNumber VARCHAR(15),  
 ShippingAddress VARCHAR(255),  
 BillingAddress VARCHAR(255)  
);

## 1.5 Orders Table

The Orders table stores order-related details, including the customer who placed the order, the order date, status, and total amount.

SQL Code:  
CREATE TABLE Orders (  
 OrderID INT PRIMARY KEY,  
 CustomerID INT,  
 OrderDate DATETIME NOT NULL,  
 OrderStatus VARCHAR(50) NOT NULL, -- e.g., Pending, Shipped, Completed  
 TotalAmount DECIMAL(10, 2) NOT NULL,  
 PaymentStatus VARCHAR(50), -- e.g., Paid, Unpaid  
 PaymentMethod VARCHAR(50), -- e.g., Credit Card, PayPal  
 ShippingAddress VARCHAR(255),  
 BillingAddress VARCHAR(255),  
 CONSTRAINT FK\_Customer FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
);

## 1.6 OrderDetails Table

The OrderDetails table stores each item in an order, including quantity and price.

SQL Code:  
CREATE TABLE OrderDetails (  
 OrderDetailID INT PRIMARY KEY,  
 OrderID INT,  
 ProductID INT,  
 Quantity INT NOT NULL,  
 UnitPrice DECIMAL(10, 2) NOT NULL,  
 CONSTRAINT FK\_Order FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),  
 CONSTRAINT FK\_Product FOREIGN KEY (ProductID) REFERENCES Products(ProductID)  
);

## 1.7 ShoppingCart Table

The ShoppingCart table manages the items customers have added to their cart before the final purchase.

SQL Code:  
CREATE TABLE ShoppingCart (  
 CartID INT PRIMARY KEY,  
 CustomerID INT,  
 ProductID INT,  
 Quantity INT NOT NULL,  
 DateAdded DATETIME NOT NULL,  
 CONSTRAINT FK\_Customer\_Cart FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),  
 CONSTRAINT FK\_Product\_Cart FOREIGN KEY (ProductID) REFERENCES Products(ProductID)  
);

## 1.8 Payments Table

The Payments table stores payment information related to completed orders.

SQL Code:  
CREATE TABLE Payments (  
 PaymentID INT PRIMARY KEY,  
 OrderID INT,  
 PaymentDate DATETIME NOT NULL,  
 PaymentAmount DECIMAL(10, 2) NOT NULL,  
 PaymentMethod VARCHAR(50), -- e.g., Credit Card, PayPal  
 PaymentStatus VARCHAR(50), -- e.g., Paid, Refunded  
 CONSTRAINT FK\_Order\_Payment FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)  
);

# 2. SQL Queries for Common Operations

## 2.1 Total Sales by Product

This query calculates the total sales revenue by product.

SQL Code:  
SELECT   
 p.ProductName,  
 SUM(od.Quantity \* od.UnitPrice) AS TotalSales  
FROM   
 OrderDetails od  
JOIN   
 Products p ON od.ProductID = p.ProductID  
GROUP BY   
 p.ProductName;

## 2.2 Sales by Category

This query calculates the total sales for each product category.

SQL Code:  
SELECT   
 c.CategoryName,  
 SUM(od.Quantity \* od.UnitPrice) AS TotalSales  
FROM   
 OrderDetails od  
JOIN   
 Products p ON od.ProductID = p.ProductID  
JOIN   
 ProductCategories c ON p.CategoryID = c.CategoryID  
GROUP BY   
 c.CategoryName;

## 2.3 Top Customers by Sales

This query returns the top customers by total sales amount.

SQL Code:  
SELECT   
 cu.FirstName,  
 cu.LastName,  
 SUM(od.Quantity \* od.UnitPrice) AS TotalSpent  
FROM   
 OrderDetails od  
JOIN   
 Orders o ON od.OrderID = o.OrderID  
JOIN   
 Customers cu ON o.CustomerID = cu.CustomerID  
GROUP BY   
 cu.FirstName, cu.LastName  
ORDER BY   
 TotalSpent DESC;

## 2.4 Inventory Status

This query returns the current stock of each product.

SQL Code:  
SELECT   
 p.ProductName,  
 p.StockQuantity,  
 SUM(od.Quantity) AS TotalSold,  
 (p.StockQuantity - SUM(od.Quantity)) AS StockRemaining  
FROM   
 Products p  
LEFT JOIN   
 OrderDetails od ON p.ProductID = od.ProductID  
GROUP BY   
 p.ProductName, p.StockQuantity;