E-Commerce Platform Database Documentation

Table of Contents

Database Overview

Database Configuration

Property	Value
Database Name	EcommerceDB
Server	BHAVADEESHREDDY\SQLEXPRESS
Database Management System	Microsoft SQL Server
Status	Production Ready
Connection Method	Entity Framework Core

Core Tables Structure

Table Name	Primary Purpose	Key Features
Users	Authentication & User Management	Role-based access, Profile data
Categories	Product Organization	Hierarchical categorization
Products	Product Catalog	Inventory tracking, Pricing
Cart	Shopping Cart Sessions	User-specific cart management
CartItems	Cart Item Details	Quantity and pricing per item
Orders	Order Management	Transaction tracking, Status updates
OrderItems	Order Line Items	Detailed order breakdown

Database Relationship

Database Schema

Users Table

```
CREATE TABLE Users (
    UserID INT IDENTITY(1,1) PRIMARY KEY,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    Email NVARCHAR(100) UNIQUE NOT
NULL,     PasswordHash NVARCHAR(255)
NOT NULL,
    PhoneNumber NVARCHAR(15),
    Role NVARCHAR(20) DEFAULT 'User' CHECK (Role IN ('Guest', 'User', 'Admin')),
    IsActive BIT DEFAULT 1,
    CreatedDate DATETIME2 DEFAULT GETDATE(),
    Address NVARCHAR(255),
    City NVARCHAR(50),
    State NVARCHAR(50),
    ZipCode NVARCHAR(10)
);
```

Categories Table

```
CREATE TABLE Categories (
    CategoryID INT IDENTITY(1,1) PRIMARY KEY,
    CategoryName NVARCHAR(100) NOT NULL,
    Description NVARCHAR(500),
    IsActive BIT DEFAULT 1,
    CreatedDate DATETIME2 DEFAULT GETDATE()
);
```

Products Table

```
CREATE TABLE Products (
    ProductID INT IDENTITY(1,1) PRIMARY KEY,
    ProductName NVARCHAR(200) NOT NULL,
    Description NVARCHAR(1000),
    Price DECIMAL(10,2) NOT NULL,
    StockQuantity INT NOT NULL DEFAULT 0,
    CategoryID INT,
    ImageURL NVARCHAR(500),
    IsActive BIT DEFAULT 1,
    CreatedDate DATETIME2 DEFAULT GETDATE(),
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID) );
```

Cart Table

```
CREATE TABLE Cart (
    CartID INT IDENTITY(1,1) PRIMARY KEY,
    UserID INT NOT NULL,
    CreatedDate DATETIME2 DEFAULT GETDATE(),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

CartItems Table

```
CREATE TABLE CartItems (
    CartItemID INT IDENTITY(1,1) PRIMARY KEY,
    CartID INT NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT NOT NULL DEFAULT 1,
    Price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (CartID) REFERENCES Cart(CartID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID) );
```

Orders Table

```
CREATE TABLE Orders (     OrderID INT
IDENTITY(1,1) PRIMARY KEY,     UserID
INT NOT NULL,
    OrderDate DATETIME2 DEFAULT GETDATE(),
    TotalAmount DECIMAL(10,2) NOT NULL,
    Status NVARCHAR(20) DEFAULT 'Pending' CHECK (Status IN ('Pending', 'Processing', 'Shipped', 'Delivered', 'Cancelled')),
    ShippingAddress NVARCHAR(500),
    PaymentMethod NVARCHAR(50),
    PaymentStatus NVARCHAR(20) DEFAULT 'Pending',
FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

OrderItems Table

```
CREATE TABLE OrderItems (
    OrderItemID INT IDENTITY(1,1) PRIMARY KEY,
    OrderID INT NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT NOT NULL,
    UnitPrice DECIMAL(10,2) NOT NULL,
    TotalPrice DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID) );
```

Sample Queries

Product Management Queries

Get All Active Products with Categories

```
SELECT p.ProductID, p.ProductName, p.Price, p.StockQuantity, c.CategoryName
FROM Products p LEFT JOIN Categories c ON
p.CategoryID = c.CategoryID
WHERE p.IsActive = 1;
```

Search Products by Name or Category

```
SELECT p.ProductID, p.ProductName, p.Price, c.CategoryName
FROM Products p LEFT JOIN Categories c ON
p.CategoryID = c.CategoryID
WHERE p.IsActive = 1
AND (p.ProductName LIKE '%search_term%' OR c.CategoryName LIKE '%search_term%');
```

Order Management Queries

Get User Order History

```
SELECT o.OrderID, o.OrderDate, o.TotalAmount, o.Status,
     u.FirstName + ' ' + u.LastName AS CustomerName
FROM Orders o
JOIN Users u ON o.UserID = u.UserID
ORDER BY o.OrderDate DESC;
```

Get Order Details with Products

```
SELECT o.OrderID, oi.ProductID, p.ProductName, oi.Quantity, oi.UnitPrice, oi.TotalPrice
FROM Orders o
JOIN OrderItems oi ON o.OrderID = oi.OrderID
JOIN Products p ON oi.ProductID = p.ProductID
WHERE o.OrderID = @OrderID;
```

Shopping Cart Queries

Get Cart Items for User

```
SELECT c.CartID, ci.ProductID, p.ProductName, ci.Quantity, ci.Price,      (ci.Quantity * ci.Price) AS TotalPrice
FROM Cart c
JOIN CartItems ci ON c.CartID = ci.CartID
JOIN Products p ON ci.ProductID = p.ProductID
WHERE c.UserID = @UserID;
```

Calculate Cart Total

```
SELECT SUM(ci.Quantity * ci.Price) AS CartTotal
FROM Cart c
JOIN CartItems ci ON c.CartID = ci.CartID
WHERE c.UserID = @UserID;
```

Analytics Queries

Sales Report by Category

```
SELECT c.CategoryName, COUNT(oi.ProductID) AS ProductsSold,
       SUM(oi.TotalPrice) AS TotalRevenue
FROM Categories c
JOIN Products p ON c.CategoryID = p.CategoryID
JOIN OrderItems oi ON p.ProductID = oi.ProductID
JOIN Orders o ON oi.OrderID = o.OrderID
WHERE o.Status = 'Delivered'
GROUP BY c.CategoryName;
```

Top Selling Products

```
SELECT TOP 10 p.ProductName, SUM(oi.Quantity) AS TotalSold,
       SUM(oi.TotalPrice) AS Revenue
FROM Products p
JOIN OrderItems oi ON p.ProductID = oi.ProductID
JOIN Orders o ON oi.OrderID = o.OrderID
WHERE o.Status = 'Delivered' GROUP
BY p.ProductID, p.ProductName
ORDER BY TotalSold DESC;
```

Review System Queries

Get Product Reviews with User Details

```
SELECT p.ProductName, r.Rating, r.Comment, r.ReviewDate,
       u.FirstName + ' ' + u.LastName AS ReviewerName
FROM Reviews r
JOIN Products p ON r.ProductID = p.ProductID
JOIN Users u ON r.UserID = u.UserID
ORDER BY r.ReviewDate DESC;
```

Get Average Rating for Product

```
SELECT p.ProductID, p.ProductName,
       AVG(CAST(r.Rating AS FLOAT)) AS AverageRating,
       COUNT(r.ReviewID) AS ReviewCount
FROM Products p
LEFT JOIN Reviews r ON p.ProductID = r.ProductID
GROUP BY p.ProductID, p.ProductName;
```

Stored Procedures

Cart Management Procedures

Add Product to Cart

```
CREATE OR ALTER PROCEDURE sp_AddToCart
   @UserID INT,
   @ProductID INT,
   @Quantity INT
AS
BEGIN
   DECLARE @CartID INT;
   DECLARE @ExistingQuantity INT = 0;

   -- Get or create cart for user
   SELECT @CartID = CartID FROM Cart WHERE UserID =
@UserID;
   IF @CartID IS NULL
   BEGIN
      INSERT INTO Cart (UserID) VALUES (@UserID);
      SET @CartID = SCOPE_IDENTITY();
   END

   -- Check if product already in cart
   SELECT @ExistingQuantity = Quantity FROM CartItems
   WHERE CartID = @CartID AND ProductID = @ProductID;

   IF @ExistingQuantity > 0
   BEGIN
      -- Update existing item
      UPDATE CartItems
      SET Quantity = Quantity + @Quantity        WHERE CartID
= @CartID AND ProductID = @ProductID;
   END
   ELSE
   BEGIN
      -- Add new item
      DECLARE @Price DECIMAL(10,2);
      SELECT @Price = Price FROM Products WHERE ProductID = @ProductID;
      INSERT INTO CartItems (CartID, ProductID, Quantity, Price)
      VALUES (@CartID, @ProductID, @Quantity, @Price);
   END
END;
```

Remove Product from Cart

```
CREATE OR ALTER PROCEDURE sp_RemoveFromCart
   @UserID INT,
   @ProductID INT
AS
BEGIN    DECLARE
@CartID INT;


   SELECT @CartID = CartID FROM Cart WHERE UserID =
@UserID;
   IF @CartID IS NOT NULL
   BEGIN
      DELETE FROM CartItems
      WHERE CartID = @CartID AND ProductID = @ProductID;
   END
END;
```

Order Processing Procedures

Process Order

Update Order Status

```
CREATE OR ALTER PROCEDURE sp_UpdateOrderStatus
   @OrderID INT,
   @Status NVARCHAR(20)
AS
BEGIN
   UPDATE Orders
   SET Status = @Status
   WHERE OrderID = @OrderID;
END;
```

User Management Procedures

Get User Profile

```
CREATE OR ALTER PROCEDURE sp_GetUserProfile
   @UserID INT
AS
BEGIN
   SELECT UserID, FirstName, LastName, Email, PhoneNumber,
        Address, City, State, ZipCode, Role, CreatedDate
   FROM Users
   WHERE UserID = @UserID AND IsActive = 1;
END;
```

Update User Profile

```
CREATE OR ALTER PROCEDURE sp_UpdateUserProfile
    @UserID INT,
    @FirstName NVARCHAR(50),
    @LastName NVARCHAR(50),
    @PhoneNumber NVARCHAR(15),
    @Address NVARCHAR(255),
    @City NVARCHAR(50),
    @State NVARCHAR(50),
    @ZipCode NVARCHAR(10)
AS
BEGIN
    UPDATE Users
    SET FirstName = @FirstName,
        LastName = @LastName,
        PhoneNumber = @PhoneNumber,
        Address = @Address,
        City = @City,
        State = @State,
        ZipCode = @ZipCode
    WHERE UserID = @UserID;
END;
```

Database Views

Product Catalog View

```
CREATE OR ALTER VIEW vw_ProductCatalog AS
SELECT
    p.ProductID,
    p.ProductName,
    p.Description,
    p.Price,
    p.StockQuantity,
    p.ImageURL,
    c.CategoryName,
    AVG(CAST(r.Rating AS FLOAT)) AS AverageRating,     COUNT(r.ReviewID) AS ReviewCount
FROM Products p
LEFT JOIN Categories c ON p.CategoryID = c.CategoryID
LEFT JOIN Reviews r ON p.ProductID = r.ProductID
WHERE p.IsActive = 1
GROUP BY p.ProductID, p.ProductName, p.Description, p.Price,
         p.StockQuantity, p.ImageURL, c.CategoryName;
```

Order Summary View

```
CREATE OR ALTER VIEW vw_OrderSummary AS
SELECT
    o.OrderID,
    o.OrderDate,
    o.TotalAmount,
    o.Status,
    u.FirstName + ' ' + u.LastName AS CustomerName,
    u.Email,
    COUNT(oi.OrderItemID) AS ItemCount
FROM Orders o
JOIN Users u ON o.UserID = u.UserID
LEFT JOIN OrderItems oi ON o.OrderID = oi.OrderID
GROUP BY o.OrderID, o.OrderDate, o.TotalAmount, o.Status,
         u.FirstName, u.LastName, u.Email;
```

Sales Analytics View

```
CREATE OR ALTER VIEW vw_SalesAnalytics AS
SELECT
    p.ProductID,
    p.ProductName,
    c.CategoryName,
    SUM(oi.Quantity) AS TotalSold,
    SUM(oi.TotalPrice) AS TotalRevenue,
    AVG(oi.UnitPrice) AS AveragePrice
FROM Products p
JOIN Categories c ON p.CategoryID = p.CategoryID
JOIN OrderItems oi ON p.ProductID = oi.ProductID
JOIN Orders o ON oi.OrderID = o.OrderID
WHERE o.Status = 'Delivered'
GROUP BY p.ProductID, p.ProductName, c.CategoryName;
```

User Activity View

```
CREATE OR ALTER VIEW vw_UserActivity AS
SELECT
    u.UserID,
    u.FirstName + ' ' + u.LastName AS CustomerName,
    u.Email,
    COUNT(DISTINCT o.OrderID) AS TotalOrders,
    SUM(o.TotalAmount) AS TotalSpent,
    COUNT(DISTINCT r.ReviewID) AS ReviewsGiven,
    MAX(o.OrderDate) AS LastOrderDate
FROM Users u
LEFT JOIN Orders o ON u.UserID = o.UserID
LEFT JOIN Reviews r ON u.UserID = r.UserID
WHERE u.IsActive = 1
GROUP BY u.UserID, u.FirstName, u.LastName, u.Email;
```

Sample Data Structure

Categories Sample Data

```
INSERT INTO Categories (CategoryName, Description) VALUES ('Electronics', 'Electronic devices and gadgets'),
('Clothing', 'Fashion and apparel'),
('Books', 'Books and literature'),
('Home & Garden', 'Home improvement and gardening'),
('Sports', 'Sports equipment and accessories'),
('Beauty', 'Beauty and personal care products');
```

Users Sample Data

```
INSERT INTO Users (FirstName, LastName, Email, PasswordHash, Role, Address, City, State, ZipCode) VALUES
('John', 'Doe', 'john@example.com', 'hashed_password_123', 'User', '123 Main St', 'Hyderabad', 'Telangana', '500001'),
('Jane', 'Smith', 'jane@example.com', 'hashed_password_456', 'User', '456 Oak Ave', 'Mumbai', 'Maharashtra', '400001'), ('Admin', 'User',
'admin@ecommerce.com', 'hashed_password_789', 'Admin', '789 Admin Blvd', 'Bangalore', 'Karnataka', '560001');
```

Products Sample Data

```
INSERT INTO Products (ProductName, Description, Price, StockQuantity, CategoryID, ImageURL) VALUES
('Laptop', 'High-performance laptop for professionals', 999.99, 50, 1, '/images/laptop.jpg'),
('Smartphone', 'Latest model smartphone with advanced features', 699.99, 75, 1, '/images/smartphone.jpg'),
('T-Shirt', 'Premium cotton t-shirt available in multiple colors', 19.99, 100, 2, '/images/tshirt.jpg'),
('Programming Book', 'Complete guide to modern programming languages', 39.99, 25, 3, '/images/book.jpg'), ('Running Shoes', 'Professional
running shoes for athletes', 129.99, 60, 5, '/images/shoes.jpg');
```

Orders Sample Data

```
INSERT INTO Orders (UserID, TotalAmount, Status, ShippingAddress, PaymentMethod) VALUES
(1, 1059.97, 'Delivered', '123 Main St, Hyderabad, Telangana 500001', 'Credit Card'),
(2, 39.99, 'Processing', '456 Oak Ave, Mumbai, Maharashtra 400001', 'PayPal'),
(1, 149.98, 'Shipped', '123 Main St, Hyderabad, Telangana 500001', 'Debit Card');
```

Database Features Summary

Security Features

- Password hashing for user authentication
- Role-based access control (Guest, User, Admin)
- Data validation through CHECK constraints
- Foreign key relationships for referential integrity

Input validation to prevent SQL injection

Performance Features

- Primary key indexing on all tables
- Foreign key indexing for join optimization
- Optimized views for common queries
- Stored procedures for complex operations
- Proper database normalization

Business Features

- Complete user management system
- Product catalog with categorization
- Shopping cart functionality
- Order processing and tracking
- Customer review and rating system
- Sales analytics and reporting

Technical Specifications

- **Database Server**: BHAVADEESHREDDY\SQLEXPRESS
- **Database Name**: EcommerceDB
- **Connection Method**: Entity Framework Core
- **Status**: Production Ready with Live Data