Web.Net Training

Data Base Project

Cosmetics Online Store System Design Document

Name: Reem Nael Abd-Alhadi

Trainer :Jaser Al-Shaer

3/3/2025

Introduction:

The Cosmetics Online Store system is designed to provide a comprehensive platform for managing an online cosmetics store. The system will allow customers to browse products, place orders, and manage their wishlists. Admins will have the ability to manage products, categories, brands, and discounts, as well as process customer requests. The system will also generate reports on sales and product performance.

1- Scope of the Project:

The project involves designing and developing a Cosmetics Online Store system. The system will allow administrators to manage products, categories, brands, ingredients, discounts, orders, and customer interactions. Customers can browse products, place orders, manage wishlists, submit product requests, and rate purchased items.

2- Mandatory Objects :

a- Admin
b- Customer
c- Products

3- Software Type:

The system will be a customized software solution tailored to the specific needs of a cosmetics online store.

4- System Objects (Abstraction) :

Entity	**Attribute**
Base Entity	Id, CreatedBy, UpdatedBy, IsActive, UpdateDate, CreationDate
Admin	FullName, Email, Password
Customer	FullName, Email, Password, ProfileImage, PhoneNumber
Product	Name, Description, CategoryId, BrandId, Quantity, Cost, Price, HasDiscount, DiscountAmount
Category	Name
Brand	Name
Ingredient	Name, Quantity
Order	CustomerId, OrderDate, TotalAmount
OrderItem	OrderId, ProductId, Quantity, Price
Wishlist	CustomerId, ProductId
Discount	ProductId, CategoryId, DiscountAmount, StartDate, EndDate
ProductRequest	CustomerId, ProductName, Description, Status
Rating	OrderItemId, Rating, Comment
IngredientProduct	IngredientProductId , Quantity

5- Object Relationships

a- Inheritance :

- Combine everything into one table (Person).
- Create separate child tables and include shared attributes in each.
- Create a parent table (Person) and child tables (Admin, Customer) with relationships.

b- **Composition:**
- Product
- Ingredient

6- System Functionalities :

a- Admin Account Should be available before anything
b- Admin Will log in to the system
c- Admin should add , edit, and deactivate the Products
d- Admin should explore products by name, category, brand, price, and discount status.
e- Admin can add, edit, delete categories
f- Admin can explore categories
g- Admin can add , edit and delete brands
h- Admin can explore the brands
i- Admin can add, edit, and delete ingredients
j- Admin can Explore Ingredients
k- Admin should be explore customers by name, email, phone number , and profile image.
l- Admin should be activate/Deactivate customer account
m- Admin should explore order by customer, order, and total amount .
n- Admin should view order details (Products, quantities, prices, total amount)
o- Admin should explore wishlist by customer and product details
p- Admin should apply or update discount for specific products or categories
q- Admin should set discount duration (start date , end date)
r- Admin should explore the approve/reject product requests submitted by customer
s- Admin should explore product rating and customer comment
t- Admin should generate sales reports, including itemized sales and total revenue.
u- Customer should Create an account with full name, email, password, profile image, and phone number.
v- Customer should Log in and log out of the system
w- Customer should Explore products by name, category, brand, price, and discount status.
x- Customer should View product details (description, ingredients, price, discounts).
y- Customer should Place orders by adding products to the cart and proceeding to checkout.
z- Customer should View purchase history and order details.
aa- Customer should Add products to the wishlist.
bb- Customer should Explore the wishlist.
cc- Customer should Submit product requests to the admin with product name and description.
dd- Customer should Rate purchased products and leave comments.

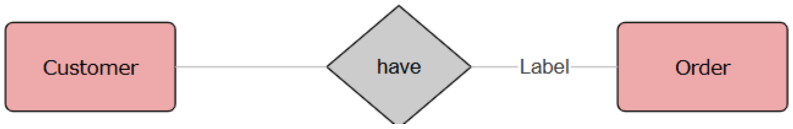
7- Entity-Relationship Diagram(ERD)

1) **Entities**
2) **Attributes**
3) **Inheritance:**
   Admin + Customer -------→ Person
4) **Composition :**
   - Product
   - Ingredient

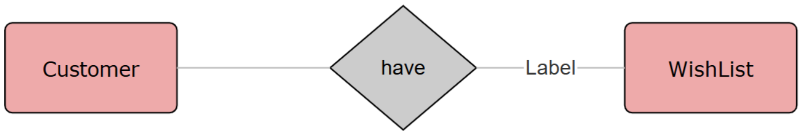
5) **Association: (many to many relationship)**

- **OrderItem (M-M)**
- **Orders (M-M)**
- **Products (M-M)**

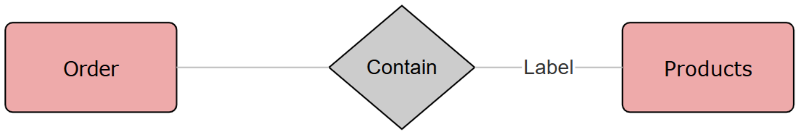
- **Customer can have many Orders (1-M)**



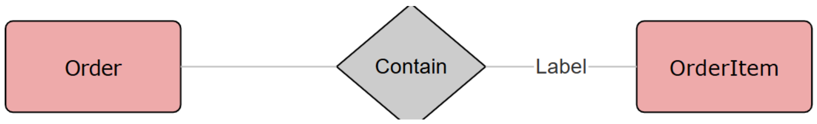
- **One Customer** can have **many Wishlist items (1-M)**



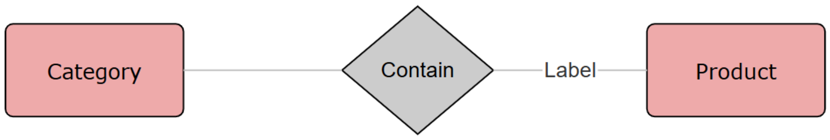
- **One order can contain multiple products(1-M)**



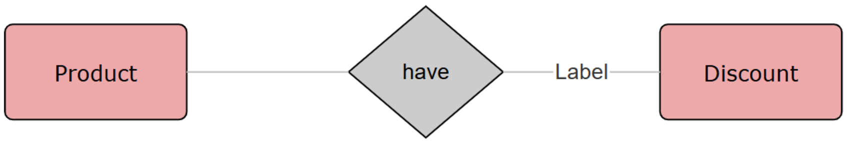
- **One order can have multiple Order Item(1-M)**



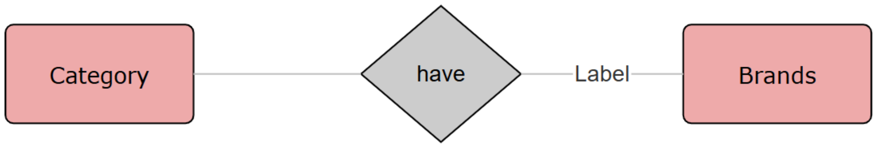
- **One category can have multiple product (1-M)**



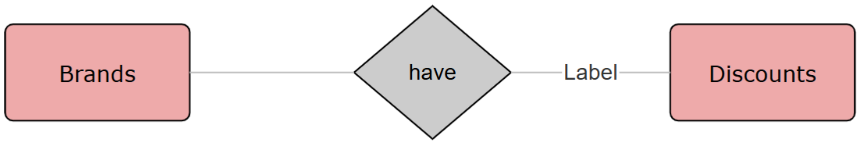
- **One Product can have many Discounts (1-M)**



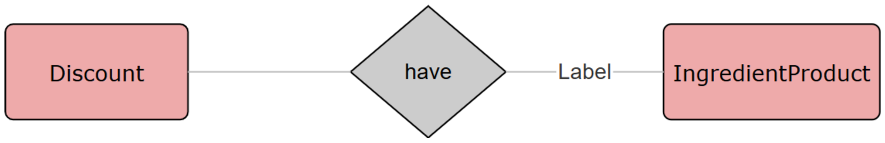
- **Many Categories can have many Brands (M-M)**



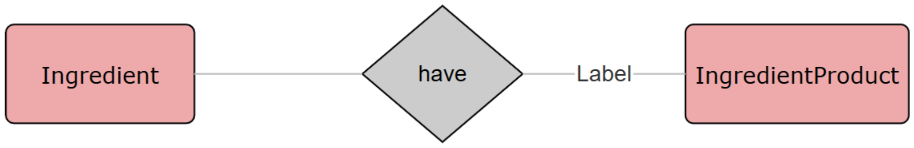
- **Many brand can have multiple discounts(M-M)**



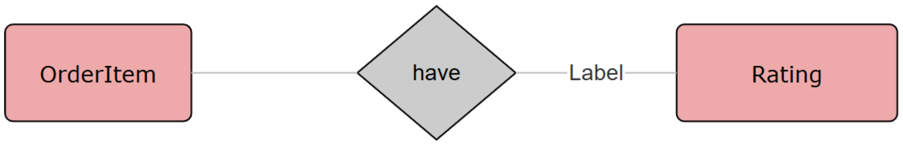
- **Many Discounts can apply to many IngredientProducts**



- **Many IngredientProduct have multiple Ingredient**



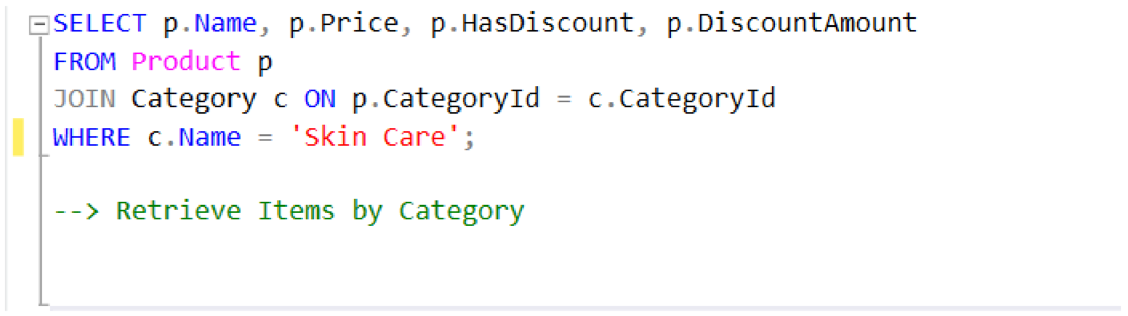
- **Each Order Item can have one Rating associated with it**



8- Database Queries :

A) Retrieve Items by Category:

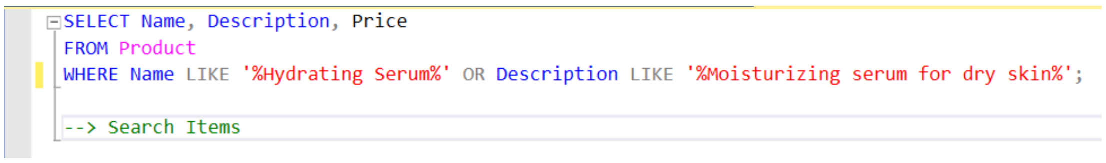
Write a query to fetch all items based on a selected category, including details such as name, price, and discount status.



```
SELECT p.Name, p.Price, p.HasDiscount, p.DiscountAmount
FROM Product p
JOIN Category c ON p.CategoryId = c.CategoryId
WHERE c.Name = 'Skin Care';

--> Retrieve Items by Category
```

B) 2. Search Items:

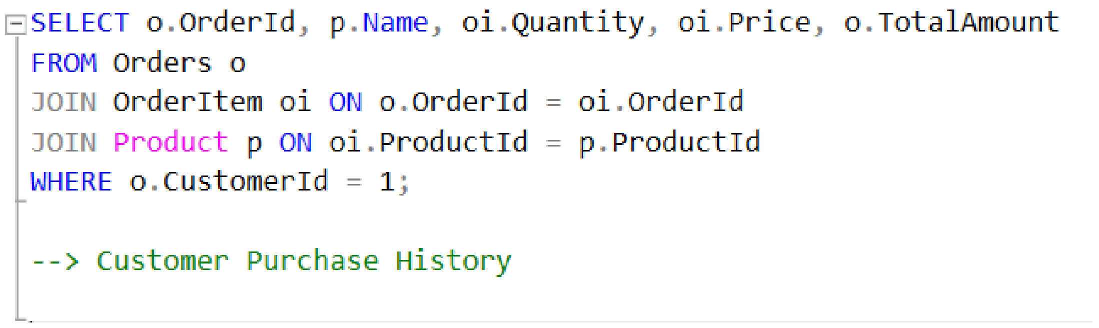
Implement a query to search for items by name or description using a keyword.



```sql
SELECT Name, Description, Price
FROM Product
WHERE Name LIKE '%Hydrating Serum%' OR Description LIKE '%Moisturizing serum for dry skin%';

--> Search Items
```

C) Customer Purchase History:

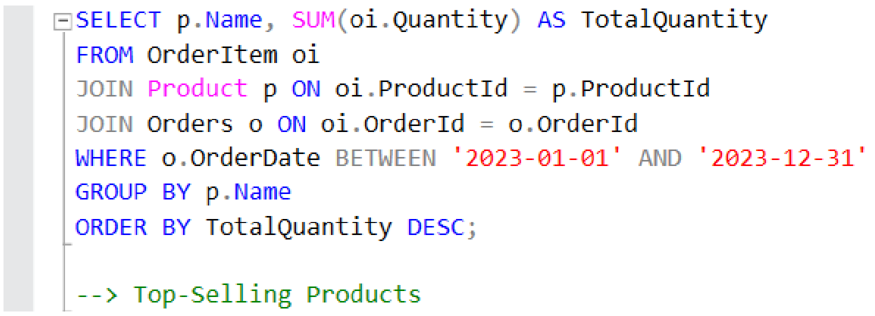
Create a query to fetch the purchase history of a specific customer, including item details, quantities, and total spent.



```sql
SELECT o.OrderId, p.Name, oi.Quantity, oi.Price, o.TotalAmount
FROM Orders o
JOIN OrderItem oi ON o.OrderId = oi.OrderId
JOIN Product p ON oi.ProductId = p.ProductId
WHERE o.CustomerId = 1;

--> Customer Purchase History
```

D) Top-Selling Products:

Write a query to identify the top-selling products based on order quantities within a specified timeframe.

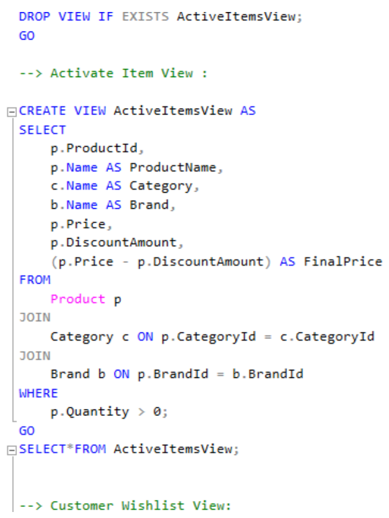


```sql
SELECT p.Name, SUM(oi.Quantity) AS TotalQuantity
FROM OrderItem oi
JOIN Product p ON oi.ProductId = p.ProductId
JOIN Orders o ON oi.OrderId = o.OrderId
WHERE o.OrderDate BETWEEN '2023-01-01' AND '2023-12-31'
GROUP BY p.Name
ORDER BY TotalQuantity DESC;

--> Top-Selling Products
```

9- Database Views

A) Active Items View:

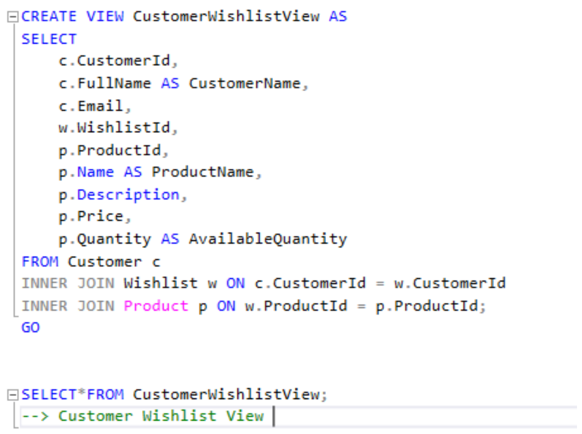
Create a view to display all items currently available for sale, including category, brand, price, and discount amount.



```sql
DROP VIEW IF EXISTS ActiveItemsView;
GO

--> Activate Item View :

CREATE VIEW ActiveItemsView AS
SELECT
    p.ProductId,
    p.Name AS ProductName,
    c.Name AS Category,
    b.Name AS Brand,
    p.Price,
    p.DiscountAmount,
    (p.Price - p.DiscountAmount) AS FinalPrice
FROM
    Product p
JOIN
    Category c ON p.CategoryId = c.CategoryId
JOIN
    Brand b ON p.BrandId = b.BrandId
WHERE
    p.Quantity > 0;
GO
SELECT*FROM ActiveItemsView;


--> Customer Wishlist View:
```

B) **Customer Wishlist View:**

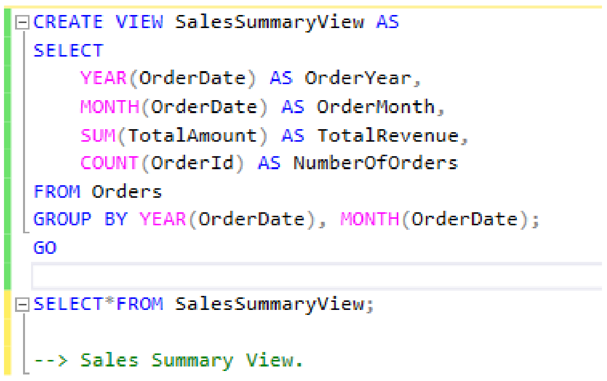
Implement a view to show all wishlist items for each customer, including item details and quantities.



```
CREATE VIEW CustomerWishlistView AS
SELECT
    c.CustomerId,
    c.FullName AS CustomerName,
    c.Email,
    w.WishlistId,
    p.ProductId,
    p.Name AS ProductName,
    p.Description,
    p.Price,
    p.Quantity AS AvailableQuantity
FROM Customer c
INNER JOIN Wishlist w ON c.CustomerId = w.CustomerId
INNER JOIN Product p ON w.ProductId = p.ProductId;
GO


SELECT*FROM CustomerWishlistView;
--> Customer Wishlist View
```

C) **Sales Summary View:**

Create a view summarizing total sales by month, including revenue and number of orders.

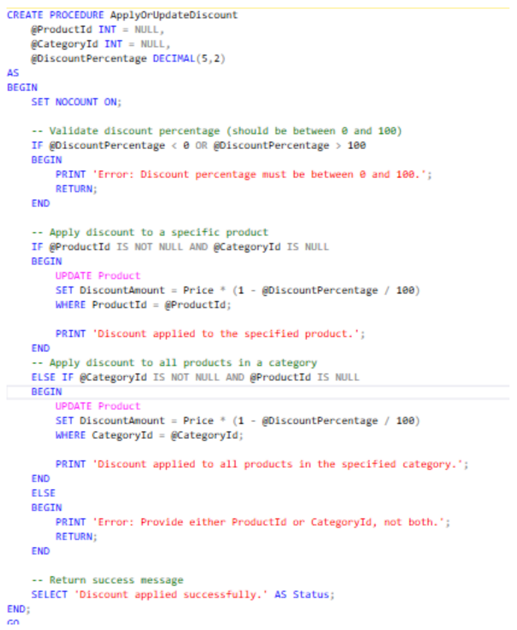


```
CREATE VIEW SalesSummaryView AS
SELECT
    YEAR(OrderDate) AS OrderYear,
    MONTH(OrderDate) AS OrderMonth,
    SUM(TotalAmount) AS TotalRevenue,
    COUNT(OrderId) AS NumberOfOrders
FROM Orders
GROUP BY YEAR(OrderDate), MONTH(OrderDate);
GO

SELECT*FROM SalesSummaryView;

--> Sales Summary View.
```

10- Stored Procedures

1. Add Item to Order:

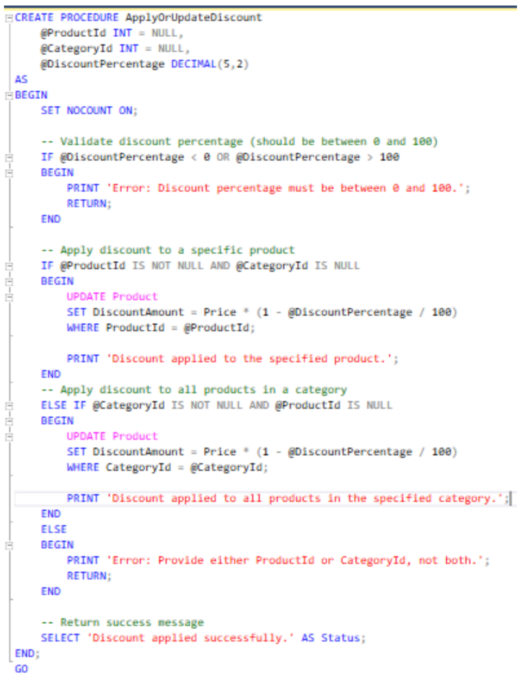
Write a stored procedure to add a selected item to a customer's order, ensuring stock availability is checked before adding.



```
CREATE PROCEDURE ApplyOrUpdateDiscount
    @ProductId INT = NULL,
    @CategoryId INT = NULL,
    @DiscountPercentage DECIMAL(5,2)
AS
BEGIN
    SET NOCOUNT ON;

    -- Validate discount percentage (should be between 0 and 100)
    IF @DiscountPercentage < 0 OR @DiscountPercentage > 100
    BEGIN
        PRINT 'Error: Discount percentage must be between 0 and 100.';
        RETURN;
    END

    -- Apply discount to a specific product
    IF @ProductId IS NOT NULL AND @CategoryId IS NULL
    BEGIN
        UPDATE Product
        SET DiscountAmount = Price * (1 - @DiscountPercentage / 100)
        WHERE ProductId = @ProductId;

        PRINT 'Discount applied to the specified product.';
    END
    -- Apply discount to all products in a category
    ELSE IF @CategoryId IS NOT NULL AND @ProductId IS NULL
    BEGIN
        UPDATE Product
        SET DiscountAmount = Price * (1 - @DiscountPercentage / 100)
        WHERE CategoryId = @CategoryId;

        PRINT 'Discount applied to all products in the specified category.';
    END
    ELSE
    BEGIN
        PRINT 'Error: Provide either ProductId or CategoryId, not both.';
        RETURN;
    END

    -- Return success message
    SELECT 'Discount applied successfully.' AS Status;
END;
GO
```

2. Manage Discounts:

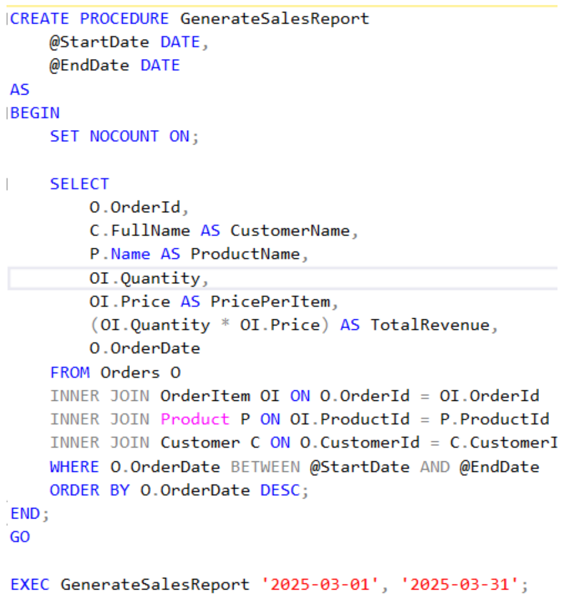
Implement a stored procedure to apply or update discounts for specific items or categories.



```sql
CREATE PROCEDURE ApplyOrUpdateDiscount
    @ProductId INT = NULL,
    @CategoryId INT = NULL,
    @DiscountPercentage DECIMAL(5,2)
AS
BEGIN
    SET NOCOUNT ON;

    -- Validate discount percentage (should be between 0 and 100)
    IF @DiscountPercentage < 0 OR @DiscountPercentage > 100
    BEGIN
        PRINT 'Error: Discount percentage must be between 0 and 100.';
        RETURN;
    END

    -- Apply discount to a specific product
    IF @ProductId IS NOT NULL AND @CategoryId IS NULL
    BEGIN
        UPDATE Product
        SET DiscountAmount = Price * (1 - @DiscountPercentage / 100)
        WHERE ProductId = @ProductId;

        PRINT 'Discount applied to the specified product.';
    END
    -- Apply discount to all products in a category
    ELSE IF @CategoryId IS NOT NULL AND @ProductId IS NULL
    BEGIN
        UPDATE Product
        SET DiscountAmount = Price * (1 - @DiscountPercentage / 100)
        WHERE CategoryId = @CategoryId;

        PRINT 'Discount applied to all products in the specified category.';
    END
    ELSE
    BEGIN
        PRINT 'Error: Provide either ProductId or CategoryId, not both.';
        RETURN;
    END

    -- Return success message
    SELECT 'Discount applied successfully.' AS Status;
END;
GO
```

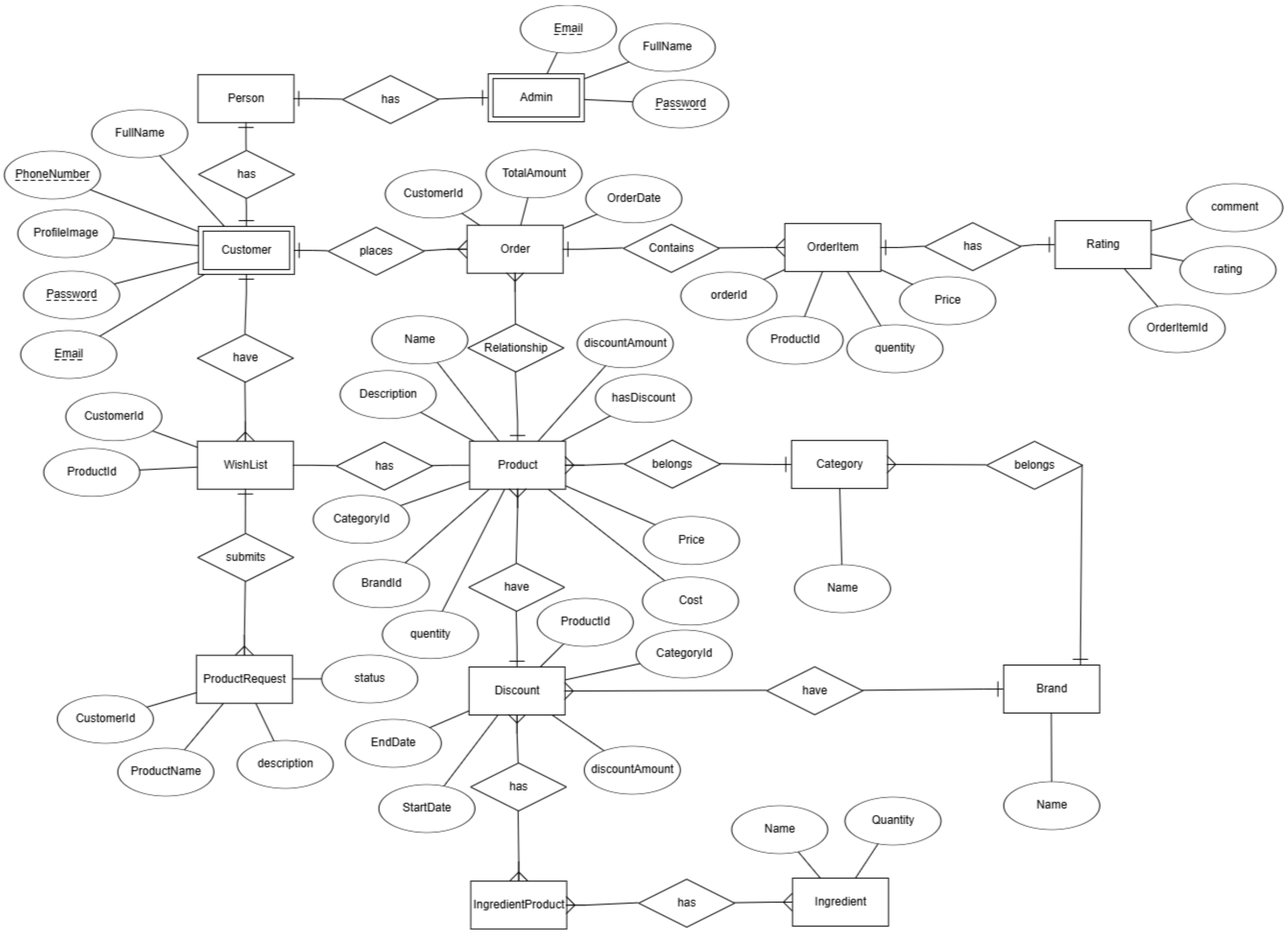
3. Generate Sales Report:

Create a stored procedure to generate a detailed sales report for a given period, including itemized sales and total revenue.



```sql
CREATE PROCEDURE GenerateSalesReport
    @StartDate DATE,
    @EndDate DATE
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        O.OrderId,
        C.FullName AS CustomerName,
        P.Name AS ProductName,
        OI.Quantity,
        OI.Price AS PricePerItem,
        (OI.Quantity * OI.Price) AS TotalRevenue,
        O.OrderDate
    FROM Orders O
    INNER JOIN OrderItem OI ON O.OrderId = OI.OrderId
    INNER JOIN Product P ON OI.ProductId = P.ProductId
    INNER JOIN Customer C ON O.CustomerId = C.CustomerI
    WHERE O.OrderDate BETWEEN @StartDate AND @EndDate
    ORDER BY O.OrderDate DESC;
END;
GO

EXEC GenerateSalesReport '2025-03-01', '2025-03-31';
```

11- ER Diagram :


Email
FullName
Password
Person
has
Admin
FullName
has
PhoneNumber
ProfileImage
Password
Email
Customer
places
CustomerId
TotalAmount
OrderDate
Order
Contains
OrderItem
has
Rating
comment
rating
OrderItemId
orderId
ProductId
quentity
Price
have
Relationship
Name
discountAmount
Description
hasDiscount
CustomerId
ProductId
WishList
has
Product
belongs
Category
belongs
CategoryId
Price
BrandId
Cost
Name
quentity
have
ProductId
CategoryId
submits
ProductRequest
status
CustomerId
ProductName
description
Discount
have
Brand
EndDate
discountAmount
StartDate
has
Name
Name
Quantity
IngredientProduct
has
Ingredient