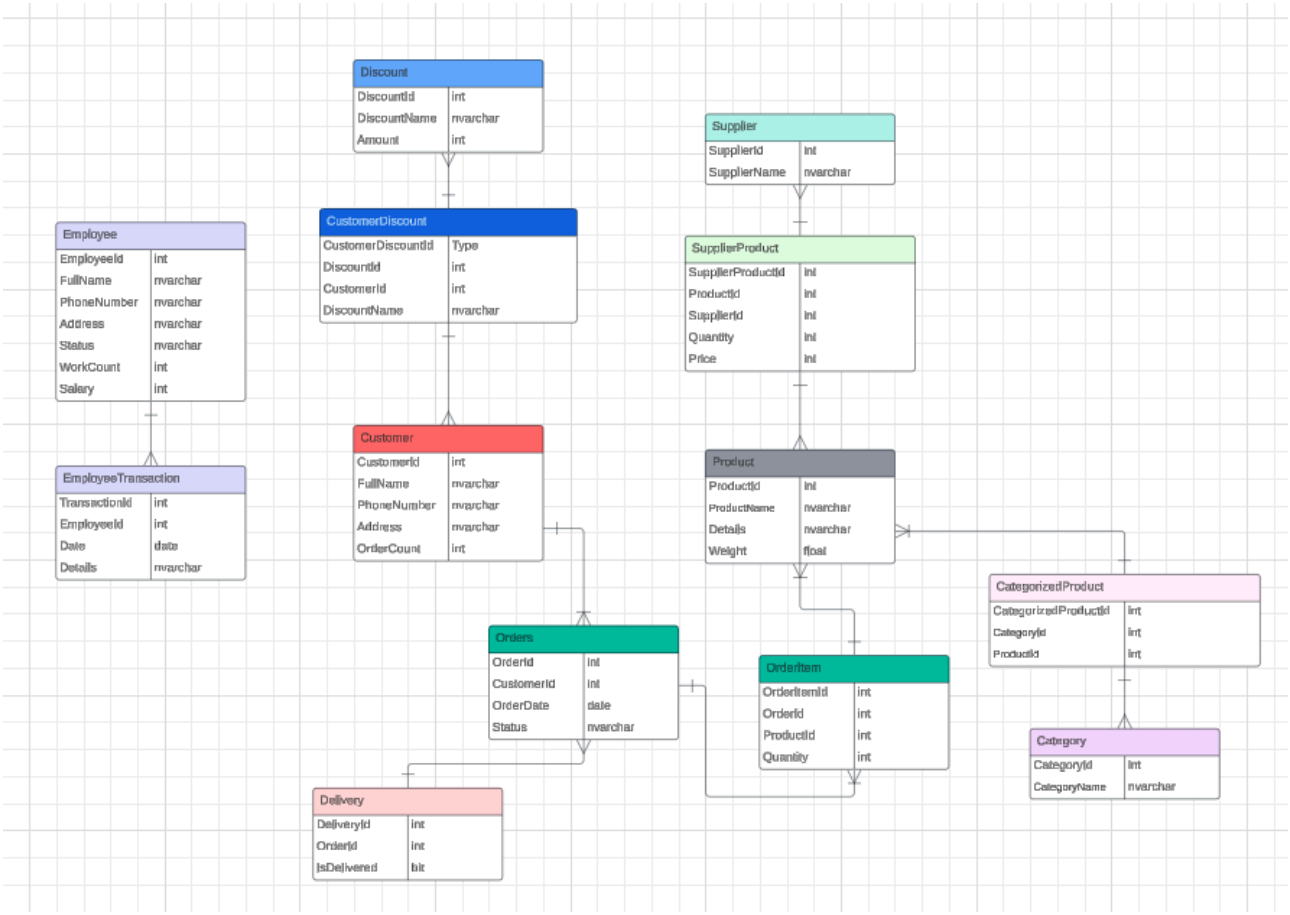
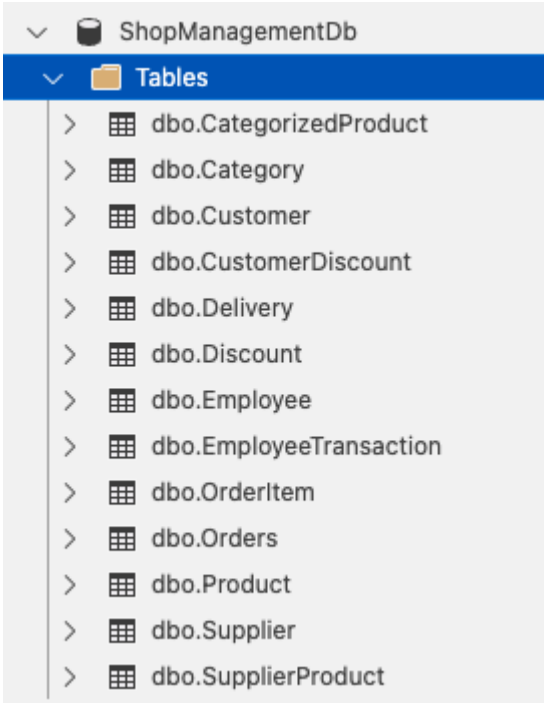
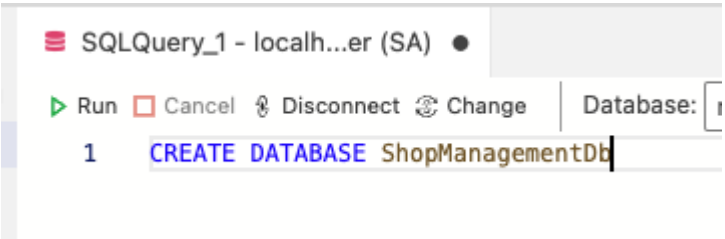
Enes Önal

Shop Management - TechCareer / SQL





1. Create Database



2. Create tables according to UML Class Diagram

```
CREATE TABLE SupplierProduct(
   SupplierProductId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   ProductId INT FOREIGN KEY REFERENCES Product(ProductId) NOT NULL,
   SupplierId INT FOREIGN KEY REFERENCES Supplier(SupplierId) NOT NULL,
   Quantity INT NOT NULL,
   Price INT NOT NULL
)

CREATE TABLE CustomerDiscount(
   CustomerDiscountId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   DiscountId INT FOREIGN KEY REFERENCES Discount(DiscountId) NOT NULL,
   CustomerId INT FOREIGN KEY REFERENCES Customer(CustomerId) NOT NULL,
   IsActive BIT NOT NULL DEFAULT 0,
)

CREATE TABLE CategorizedProduct(
   CategorizedProductId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   CategoryId INT FOREIGN KEY REFERENCES Category(CategoryId),
   ProductId INT FOREIGN KEY REFERENCES Product(ProductId)
)

CREATE TABLE Category(
   CategoryId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   CategoryName NVARCHAR(75)
)

CREATE TABLE Orders(
   OrderId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   CustomerId INT FOREIGN KEY REFERENCES Customer(CustomerId) NOT NULL,
   OrderDate DATE
   Status NVARCHAR
)
```

```sql
CREATE TABLE OrderItem(
   OrderItemId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   OrderId INT FOREIGN KEY REFERENCES Orders(OrderId) NOT NULL,
   ProductId INT FOREIGN KEY REFERENCES Product(ProductId) NOT NULL,
   Quantity INT NOT NULL,
)



CREATE TABLE Delivery(
   DeliveryId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   OrderId INT FOREIGN KEY REFERENCES Orders(OrderId) NOT NULL,
   IsDelivered BIT NOT NULL DEFAULT 0
)

CREATE TABLE EmployeeTransaction(
   EmployeeTransactionId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   EmployeeId INT FOREIGN KEY REFERENCES Employee(EmployeeId),
   TransactionDate DATE NOT NULL,
   Details NVARCHAR(200),
)

CREATE TABLE Employee(
   EmployeeId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   EmployeeName NVARCHAR(60) NOT NULL,
   PhoneNumber NVARCHAR(15),
   Address NVARCHAR(MAX),
   Status NVARCHAR(60) NOT NULL,
   Salary DECIMAL(6,2) NOT NULL
   WorkCount INT,
)

CREATE TABLE Customer(
   CustomerId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   FullName NVARCHAR(60) NOT NULL,
   PhoneNumber NVARCHAR(15) NOT NULL,
   Address NVARCHAR(MAX) NOT NULL,
   OrderCount INT
)
CREATE TABLE Supplier(
   SupplierId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   SupplierName NVARCHAR(100) NOT NULL,
)
```

```
CREATE TABLE Product(
   ProductId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   ProductName NVARCHAR(100) NOT NULL,
   ProductDetails NVARCHAR(MAX),
   Weight INT,
)

CREATE TABLE Category(
   CategoryId INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
   CategoryName NVARCHAR(60) NOT NULL,
)
```

3. Functions

- Active Discount Check Function

```
CREATE
CREATE FUNCTION dbo.CheckActiveDiscount (@CustomerId INT)
RETURNS BIT
AS
BEGIN
   DECLARE @isActive BIT;

   IF EXISTS (
       SELECT 1
       FROM CustomerDiscount
       WHERE CustomerId = @CustomerId AND IsActive = 1
   )
   BEGIN
       SET @isActive = 1;
   END
   ELSE
   BEGIN
       SET @isActive = 0;
   END

   RETURN @isActive;
END;

RUN
DECLARE @isActive BIT;

SET @isActive = dbo.CheckActiveDiscount(@CustomerId);

SELECT @isActive AS IsActive;
```

- Function to List Products by Supplier

```
CREATE
CREATE FUNCTION GetProductsBySupplier (@SupplierId INT)
RETURNS TABLE
AS
RETURN (
   SELECT p.ProductName, p.ProductDetails, p.Weight
   FROM Product p
   INNER JOIN SupplierProduct sp ON p.ProductId = sp.ProductId
   WHERE sp.SupplierId = @SupplierId
);

RUN
SELECT *
FROM dbo.GetProductsBySupplier(2);
```

- Function to Get Order Count

```
CREATE
CREATE FUNCTION GetOrderCountForCustomer (@CustomerId INT)
RETURNS INT
AS
BEGIN
   DECLARE @OrderCount INT

   SELECT @OrderCount = COUNT(*)
   FROM Orders
   WHERE CustomerId = @CustomerId

   RETURN @OrderCount
END;

RUN
DECLARE @OrderCount INT;

SET @OrderCount = dbo.GetOrderCountForCustomer(@CustomerId);
SELECT @OrderCount AS OrderCount;
```

4. Triggers

- Customer Order Count Update Trigger

```
CREATE
CREATE TRIGGER OrderInserted
ON Orders
AFTER INSERT
AS
BEGIN
   UPDATE Customer
   SET OrderCount = OrderCount + 1
   WHERE CustomerId IN (SELECT CustomerId FROM inserted);
END;

RUN
INSERT INTO Orders (CustomerId, OrderDate)
VALUES (1, GETDATE());
```

- Employee Work Count Update Trigger

```
CREATE
CREATE TRIGGER EmployeeTransactionInsert
ON EmployeeTransaction
AFTER INSERT
AS
BEGIN
   UPDATE Employee
   SET WorkCount = WorkCount + 1
   WHERE EmployeeId IN (SELECT EmployeeId FROM inserted);
END;

RUN
INSERT INTO EmployeeTransaction (EmployeeId, TransactionDate, Details)
VALUES (1, GETDATE(), 'Dump Transaction');
```

- Order Status Update Trigger When Delivery Completed

```
CREATE
CREATE TRIGGER DeliveryCompleted
ON Delivery
AFTER UPDATE
AS
BEGIN
   IF UPDATE(IsDelivered) AND EXISTS (SELECT 1 FROM inserted WHERE
IsDelivered = 1)
   BEGIN
       UPDATE Orders
       SET Status = 'Delivered'
       WHERE OrderId IN (SELECT OrderId FROM inserted WHERE IsDelivered =
1);
   END;
END;

RUN
UPDATE Delivery
SET IsDelivered = 1
WHERE DeliveryId = 1;
```