```
/* Group Members
1.Meron Kifle......GUR/01196/16
2.Hlina Shambel....GUR/01604/16
3.Natnael Zemene...GUR/02204/16
4.Eden Shewaye.....GUR/01134/16
5.Lidiya Asefa.....GUR/02780/16
6.Tilahun Misikir...GUR/02870/16 (Group Leader)
*/
--SUPERMARKET MANAGEMENT SYSTEM
-- DDL: Create Sequences and Tables

-- Sequence and Table for Categories
CREATE SEQUENCE Categories_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Categories (
   CategoryID NUMBER PRIMARY KEY,
   CategoryName VARCHAR2(50) NOT NULL,
   Description CLOB
);

CREATE OR REPLACE TRIGGER Categories_trigger
BEFORE INSERT ON Categories
FOR EACH ROW
BEGIN
   IF :new.CategoryID IS NULL THEN
      SELECT Categories_seq.NEXTVAL INTO :new.CategoryID FROM dual;
   END IF;
END;
/

-- Sequence and Table for Suppliers
CREATE SEQUENCE Suppliers_seq START WITH 1 INCREMENT BY 1;
CREATE TABLE Suppliers (
    SupplierID NUMBER PRIMARY KEY,
    SupplierName VARCHAR2(100) NOT NULL,
    ContactName VARCHAR2(50),
    Phone VARCHAR2(20),
    Email VARCHAR2(100),
    Address CLOB
);

CREATE OR REPLACE TRIGGER Suppliers_trigger
BEFORE INSERT ON Suppliers
FOR EACH ROW
BEGIN
    IF :new.SupplierID IS NULL THEN
        SELECT Suppliers_seq.NEXTVAL INTO :new.SupplierID FROM dual;
    END IF;
END;
/

-- Sequence and Table for Products
CREATE SEQUENCE Products_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Products (
    ProductID NUMBER PRIMARY KEY,
    ProductName VARCHAR2(100) NOT NULL,
    CategoryID NUMBER,
    SupplierID NUMBER,
    UnitPrice NUMBER(10, 2) NOT NULL,
    Description CLOB,
    ManufactureDate DATE,
    ExpiryDate DATE,
    CONSTRAINT fk_Category FOREIGN KEY (CategoryID) REFERENCES
Categories(CategoryID),

    CONSTRAINT fk_Supplier FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)

);


CREATE OR REPLACE TRIGGER Products_trigger

BEFORE INSERT ON Products

FOR EACH ROW

BEGIN

    IF :new.ProductID IS NULL THEN

        SELECT Products_seq.NEXTVAL INTO :new.ProductID FROM dual;

    END IF;

END;

/


-- Sequence and Table for Inventory

CREATE SEQUENCE Inventory_seq START WITH 1 INCREMENT BY 1;


CREATE TABLE Inventory (

    InventoryID NUMBER PRIMARY KEY,

    ProductID NUMBER,

    QuantityInStock NUMBER NOT NULL,

    ReorderLevel NUMBER DEFAULT 10,

    LastUpdated TIMESTAMP DEFAULT SYSTIMESTAMP,

    CONSTRAINT fk_Product_Inventory FOREIGN KEY (ProductID) REFERENCES
Products(ProductID),

    CONSTRAINT chk_Quantity CHECK (QuantityInStock >= 0)

);


CREATE OR REPLACE TRIGGER Inventory_trigger

BEFORE INSERT ON Inventory

FOR EACH ROW

BEGIN
```

```
    IF :new.InventoryID IS NULL THEN
        SELECT Inventory_seq.NEXTVAL INTO :new.InventoryID FROM dual;
    END IF;
END;
/

-- Sequence and Table for Customers
CREATE SEQUENCE Customers_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Customers (
    CustomerID NUMBER PRIMARY KEY,
    FirstName VARCHAR2(50) NOT NULL,
    LastName VARCHAR2(50) NOT NULL,
    City VARCHAR2(20),
    SubCity VARCHAR2(20),
    Kebele NUMBER,
    Email VARCHAR2(100),
    Phone VARCHAR2(20),
    Address CLOB,
    LoyaltyPoints NUMBER DEFAULT 0
);

CREATE OR REPLACE TRIGGER Customers_trigger
BEFORE INSERT ON Customers
FOR EACH ROW
BEGIN
    IF :new.CustomerID IS NULL THEN
        SELECT Customers_seq.NEXTVAL INTO :new.CustomerID FROM dual;
    END IF;
END;
/
```

```
-- Sequence and Table for Employees
CREATE SEQUENCE Employees_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Employees (
    EmployeeID NUMBER PRIMARY KEY,
    FirstName VARCHAR2(50) NOT NULL,
    LastName VARCHAR2(50) NOT NULL,
    City VARCHAR2(20),
    SubCity VARCHAR2(20),
    Kebele NUMBER,
    Salary NUMBER(10, 2) NOT NULL,
    Position VARCHAR2(30),
    Email VARCHAR2(100),
    Phone VARCHAR2(20),
    HireDate DATE NOT NULL,
    CONSTRAINT chk_Salary CHECK (Salary > 0)
);

CREATE OR REPLACE TRIGGER Employees_trigger
BEFORE INSERT ON Employees
FOR EACH ROW
BEGIN
    IF :new.EmployeeID IS NULL THEN
        SELECT Employees_seq.NEXTVAL INTO :new.EmployeeID FROM dual;
    END IF;
END;
/

-- Sequence and Table for Store
CREATE SEQUENCE Store_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Store (
    StoreID NUMBER PRIMARY KEY,
    Name VARCHAR2(100) NOT NULL,
    Address CLOB,
    Phone VARCHAR2(20),
    ManagerID NUMBER,
    OpenTime TIMESTAMP,
    CloseTime TIMESTAMP,
    CONSTRAINT fk_Manager FOREIGN KEY (ManagerID) REFERENCES
Employees(EmployeeID)
);

CREATE OR REPLACE TRIGGER Store_trigger
BEFORE INSERT ON Store
FOR EACH ROW
BEGIN
    IF :new.StoreID IS NULL THEN
        SELECT Store_seq.NEXTVAL INTO :new.StoreID FROM dual;
    END IF;
END;
/

-- Sequence and Table for Sales
CREATE SEQUENCE Sales_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Sales (
    SaleID NUMBER PRIMARY KEY,
    CustomerID NUMBER,
    EmployeeID NUMBER,
    StoreID NUMBER,
    SaleDate TIMESTAMP DEFAULT SYSTIMESTAMP,
    TotalAmount NUMBER(10, 2) NOT NULL,
    CONSTRAINT fk_Customer_Sales FOREIGN KEY (CustomerID) REFERENCES
Customers(CustomerID),
    CONSTRAINT fk_Employee_Sales FOREIGN KEY (EmployeeID) REFERENCES
Employees(EmployeeID),
    CONSTRAINT fk_Store_Sales FOREIGN KEY (StoreID) REFERENCES Store(StoreID)
);

CREATE OR REPLACE TRIGGER Sales_trigger
BEFORE INSERT ON Sales
FOR EACH ROW
BEGIN
    IF :new.SaleID IS NULL THEN
        SELECT Sales_seq.NEXTVAL INTO :new.SaleID FROM dual;
    END IF;
END;
/

-- Sequence and Table for SaleDetails
CREATE SEQUENCE SaleDetails_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE SaleDetails (
    SaleDetailID NUMBER PRIMARY KEY,
    SaleID NUMBER,
    ProductID NUMBER,
    Quantity NUMBER NOT NULL,
    UnitPrice NUMBER(10, 2) NOT NULL,
    Subtotal AS (Quantity * UnitPrice),
    CONSTRAINT fk_Sale_SaleDetails FOREIGN KEY (SaleID) REFERENCES Sales(SaleID),
    CONSTRAINT fk_Product_SaleDetails FOREIGN KEY (ProductID) REFERENCES
Products(ProductID)
);

CREATE OR REPLACE TRIGGER SaleDetails_trigger
BEFORE INSERT ON SaleDetails
FOR EACH ROW
```

```
BEGIN
    IF :new.SaleDetailID IS NULL THEN
        SELECT SaleDetails_seq.NEXTVAL INTO :new.SaleDetailID FROM dual;
    END IF;
END;
/

-- Sequence and Table for Orders
CREATE SEQUENCE Orders_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Orders (
    OrderID NUMBER PRIMARY KEY,
    SupplierID NUMBER,
    OrderDate TIMESTAMP DEFAULT SYSTIMESTAMP,
    TotalAmount NUMBER(10, 2),
    Status VARCHAR2(20) DEFAULT 'Pending',
    CONSTRAINT fk_Supplier_Orders FOREIGN KEY (SupplierID) REFERENCES
Suppliers(SupplierID),
    CONSTRAINT chk_Status CHECK (Status IN ('Pending', 'Completed', 'Cancelled'))
);

CREATE OR REPLACE TRIGGER Orders_trigger
BEFORE INSERT ON Orders
FOR EACH ROW
BEGIN
    IF :new.OrderID IS NULL THEN
        SELECT Orders_seq.NEXTVAL INTO :new.OrderID FROM dual;
    END IF;
END;
/

-- Sequence and Table for OrderDetails
CREATE SEQUENCE OrderDetails_seq START WITH 1 INCREMENT BY 1;
CREATE TABLE OrderDetails (
    OrderDetailID NUMBER PRIMARY KEY,
    OrderID NUMBER,
    ProductID NUMBER,
    Quantity NUMBER NOT NULL,
    UnitPrice NUMBER(10, 2) NOT NULL,
    Discount NUMBER(10, 2) DEFAULT 0.00,
    Subtotal AS (Quantity * UnitPrice),
    CONSTRAINT fk_Order_OrderDetails FOREIGN KEY (OrderID) REFERENCES
Orders(OrderID),
    CONSTRAINT fk_Product_OrderDetails FOREIGN KEY (ProductID) REFERENCES
Products(ProductID)
);

CREATE OR REPLACE TRIGGER OrderDetails_trigger
BEFORE INSERT ON OrderDetails
FOR EACH ROW
BEGIN
    IF :new.OrderDetailID IS NULL THEN
        SELECT OrderDetails_seq.NEXTVAL INTO :new.OrderDetailID FROM dual;
    END IF;
END;
/

-- Sequence and Table for Bill
CREATE SEQUENCE Bill_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Bill (
    BillID NUMBER PRIMARY KEY,
    SaleID NUMBER,
    BillDate TIMESTAMP,
    PaidAmount NUMBER(10, 2) NOT NULL,
    TotalPayment NUMBER(10, 2) NOT NULL,
    EmployeeID NUMBER,
    StoreID NUMBER,
    CONSTRAINT fk_Sale_Bill FOREIGN KEY (SaleID) REFERENCES Sales(SaleID),
    CONSTRAINT fk_Employee_Bill FOREIGN KEY (EmployeeID) REFERENCES
Employees(EmployeeID),
    CONSTRAINT fk_Store_Bill FOREIGN KEY (StoreID) REFERENCES Store(StoreID)
);

CREATE OR REPLACE TRIGGER Bill_trigger
BEFORE INSERT ON Bill
FOR EACH ROW
BEGIN
    IF :new.BillID IS NULL THEN
        SELECT Bill_seq.NEXTVAL INTO :new.BillID FROM dual;
    END IF;
END;
/

-- **DDL: Create Indexes for Performance**
CREATE INDEX idx_ProductName ON Products (ProductName);
CREATE INDEX idx_CustomerID ON Sales (CustomerID);
CREATE INDEX idx_SaleDate ON Sales (SaleDate);

-- Add a Discount column to Products
ALTER TABLE Products ADD Discount NUMBER(10, 2) DEFAULT 0.00;

-- DML: Insert Sample Data

-- Insert into Categories
   INSERT INTO Categories (CategoryName, Description)
   VALUES ('Groceries', 'Food and household items');
   INSERT INTO Categories (CategoryName, Description)
   VALUES ('Beverages', 'Drinks and beverages');
   INSERT  INTO Categories (CategoryName, Description)
   VALUES ('Personal Care', 'Hygiene and grooming products');

-- Insert into Suppliers
    INSERT INTO Suppliers (SupplierName, ContactName, Phone, Email, Address)
    VALUES ('AgriDistributors', 'Adane Getachew', '+251911223344', 'adaneagridist2@gmail.com', 
'Addis Ababa, Bole');
    INSERT INTO Suppliers (SupplierName, ContactName, Phone, Email, Address)
    VALUES ('FoodTech Supplies', 'Daniel Birara', '+251922334455', 'danifoodtech3@gmail.com', 
'Addis Ababa, Kirkos');
    INSERT INTO Suppliers (SupplierName, ContactName, Phone, Email, Address)
    VALUES ('DailyGoods Co.', 'Amare Zewdu', '+251933445566', 'mikedailygoods6@gmail.com', 
'Addis Ababa, Arada');

-- Insert into Products
    INSERT INTO Products (ProductName, CategoryID, SupplierID, UnitPrice, Description, 
ManufactureDate, ExpiryDate)
    VALUES ('Rice', 1, 1, 150.00, 'Basmati Rice', TO_DATE('2023-06-01', 'YYYY-MM-DD'), 
TO_DATE('2025-05-31', 'YYYY-MM-DD'));
    INSERT INTO Products (ProductName, CategoryID, SupplierID, UnitPrice, Description, 
ManufactureDate, ExpiryDate)
    VALUES ('Milk', 2, 2, 25.00, 'Lacto Milk', TO_DATE('2023-07-15', 'YYYY-MM-DD'), 
TO_DATE('2024-07-14', 'YYYY-MM-DD'));
    INSERT INTO Products (ProductName, CategoryID, SupplierID, UnitPrice, Description, 
ManufactureDate, ExpiryDate)
    VALUES ('Bread', 1, 3, 15.00, 'FreshBake Bread', TO_DATE('2023-08-20', 'YYYY-MM-DD'), 
TO_DATE('2024-08-19', 'YYYY-MM-DD'));
    INSERT INTO Products (ProductName, CategoryID, SupplierID, UnitPrice, Description, 
ManufactureDate, ExpiryDate)
    VALUES ('Sugar', 1, 1, 80.00, 'SweetCo Sugar', TO_DATE('2023-09-10', 'YYYY-MM-DD'), 
TO_DATE('2025-09-09', 'YYYY-MM-DD'));
    INSERT INTO Products (ProductName, CategoryID, SupplierID, UnitPrice, Description, 
ManufactureDate, ExpiryDate)
    VALUES ('Cooking Oil', 1, 2, 200.00, 'GoldenDrop Oil', TO_DATE('2023-10-05', 'YYYY-MM-
DD'), TO_DATE('2025-10-04', 'YYYY-MM-DD'));
```

```
-- Insert into Inventory
   INSERT INTO Inventory (ProductID, QuantityInStock, ReorderLevel)
   VALUES (1, 1000, 100);
   INSERT INTO Inventory (ProductID, QuantityInStock, ReorderLevel)
   VALUES (2, 750, 50);
   INSERT INTO Inventory (ProductID, QuantityInStock, ReorderLevel)
   VALUES (3, 300, 20);
   INSERT INTO Inventory (ProductID, QuantityInStock, ReorderLevel)
   VALUES (4, 500, 50);
   INSERT INTO Inventory (ProductID, QuantityInStock, ReorderLevel)
   VALUES (5, 200, 30);


-- Insert into Customers
   INSERT INTO Customers (FirstName, LastName, City, SubCity, Kebele, Email, Phone, Address,
LoyaltyPoints)
   VALUES ('Tewodros', 'Kassahun', 'Addis Ababa', 'Bole', 12, 'tewodros@gmail.com',
'+251911112233', 'Bole Road', 100);
   INSERT INTO Customers (FirstName, LastName, City, SubCity, Kebele, Email, Phone, Address,
LoyaltyPoints)
   VALUES ('Alemnesh', 'Girma', 'Addis Ababa', 'Kirkos', 5, 'alemnesh@gmail.com',
'+251922334455', 'Kirkos St', 50);
   INSERT INTO Customers (FirstName, LastName, City, SubCity, Kebele, Email, Phone, Address,
LoyaltyPoints)
   VALUES ('Yared', 'Getachew', 'Addis Ababa', 'Yeka', 4, 'yared@gmail.com', '+251933445566',
'Yeka Ave', 75);


-- Insert into Employees
   INSERT INTO Employees (FirstName, LastName, City, SubCity, Kebele, Salary, Position, Email,
Phone, HireDate)
   VALUES ('Abebe', 'Kebede', 'Addis Ababa', 'Bole', 12, 15000.00, 'Manager', 'abebe@gmail.com',
'+251911223344', TO_DATE('2023-01-01', 'YYYY-MM-DD'));
```

```
    INSERT INTO Employees (FirstName, LastName, City, SubCity, Kebele, Salary, Position, Email,
Phone, HireDate)

    VALUES ('Metadel', 'Werku', 'Addis Ababa', 'Kirkos', 5, 12000.00, 'Cashier',
'metadel@gmail.com', '+251922334455', TO_DATE('2023-02-01', 'YYYY-MM-DD'));

    INSERT INTO Employees (FirstName, LastName, City, SubCity, Kebele, Salary, Position, Email,
Phone, HireDate)

    VALUES ('Shegaw', 'Birhanu', 'Addis Ababa', 'Yeka', 4, 8000.00, 'Clerk', 'shegaw@gmail.com',
'+251933445566', TO_DATE('2023-03-01', 'YYYY-MM-DD'));



-- Insert into Store

    INSERT INTO Store (Name, Address, Phone, ManagerID, OpenTime, CloseTime)

    VALUES ('FreshMart', 'Addis Ababa, Bole', '+251973546201', 1, TO_TIMESTAMP('08:00:00',
'HH24:MI:SS'), TO_TIMESTAMP('20:00:00', 'HH24:MI:SS'));

    INSERT INTO Store (Name, Address, Phone, ManagerID, OpenTime, CloseTime)

    VALUES ('SuperSave', 'Addis Ababa, Kolfe Keraniyo', '+251921539151', 2,
TO_TIMESTAMP('09:00:00', 'HH24:MI:SS'), TO_TIMESTAMP('21:00:00', 'HH24:MI:SS'));

    INSERT INTO Store (Name, Address, Phone, ManagerID, OpenTime, CloseTime)

    VALUES ('DailyNeeds', 'Addis Ababa, Yeka', '+251985073484', 3, TO_TIMESTAMP('07:00:00',
'HH24:MI:SS'), TO_TIMESTAMP('22:00:00', 'HH24:MI:SS'));



-- Insert into Sales

    INSERT INTO Sales (CustomerID, EmployeeID, StoreID, SaleDate, TotalAmount)

    VALUES (1, 1, 1, TO_TIMESTAMP('2023-01-12 00:00:00', 'YYYY-MM-DD HH24:MI:SS'),
8000.00);

    INSERT INTO Sales (CustomerID, EmployeeID, StoreID, SaleDate, TotalAmount)

    VALUES (2, 2, 1, TO_TIMESTAMP('2023-02-18 00:00:00', 'YYYY-MM-DD HH24:MI:SS'),
1250.00);

    INSERT INTO Sales (CustomerID, EmployeeID, StoreID, SaleDate, TotalAmount)

    VALUES (3, 3, 2, TO_TIMESTAMP('2023-03-22 00:00:00', 'YYYY-MM-DD HH24:MI:SS'),
450.00);



-- Insert into SaleDetails

    INSERT INTO SaleDetails (SaleID, ProductID, Quantity, UnitPrice)
    VALUES (1, 1, 50, 150.00);
    INSERT INTO SaleDetails (SaleID, ProductID, Quantity, UnitPrice)
    VALUES (1, 2, 20, 25.00);
    INSERT INTO SaleDetails (SaleID, ProductID, Quantity, UnitPrice)
    VALUES (2, 3, 30, 15.00);



-- Insert into Orders
    INSERT INTO Orders (SupplierID, OrderDate, TotalAmount, Status)
    VALUES (1, TO_TIMESTAMP('2023-01-15 00:00:00', 'YYYY-MM-DD HH24:MI:SS'),
60000.00, 'Pending');
    INSERT INTO Orders (SupplierID, OrderDate, TotalAmount, Status)
    VALUES (2, TO_TIMESTAMP('2023-02-20 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 6000.00,
'Completed');
    INSERT INTO Orders (SupplierID, OrderDate, TotalAmount, Status)
    VALUES (3, TO_TIMESTAMP('2023-03-10 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1200.00,
'Pending');


-- Insert into OrderDetails
    INSERT INTO OrderDetails (OrderID, ProductID, Quantity, UnitPrice, Discount)
    VALUES (1, 1, 500, 120.00, 0.00);
    INSERT INTO OrderDetails (OrderID, ProductID, Quantity, UnitPrice, Discount)
    VALUES (1, 2, 300, 20.00, 0.00);
    INSERT INTO OrderDetails (OrderID, ProductID, Quantity, UnitPrice, Discount)
    VALUES (2, 3, 100, 12.00, 0.00);



-- Insert into Bill
    INSERT INTO Bill (SaleID, BillDate, PaidAmount, TotalPayment, EmployeeID, StoreID)
    VALUES (1, TO_TIMESTAMP('2023-01-12 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 8000.00,
8000.00, 1, 1);
    INSERT INTO Bill (SaleID, BillDate, PaidAmount, TotalPayment, EmployeeID, StoreID)
    VALUES (2, TO_TIMESTAMP('2023-02-18 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1250.00,
1250.00, 2, 1);
```

```
    INSERT INTO Bill (SaleID, BillDate, PaidAmount, TotalPayment, EmployeeID, StoreID)
    VALUES (3, TO_TIMESTAMP('2023-03-22 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 450.00,
450.00, 3, 2);


SELECT * FROM Categories;
SELECT * FROM Suppliers;
SELECT * FROM Products;
SELECT * FROM Inventory;
SELECT * FROM Customers;
SELECT * FROM Employees;
SELECT * FROM Store;
SELECT * FROM Sales;
SELECT * FROM SaleDetails;
SELECT * FROM Orders;
SELECT * FROM OrderDetails;
SELECT * FROM Bill;



-- DDL: Create Procedures


-- Procedure to add a new employee
CREATE OR REPLACE PROCEDURE AddEmployee(
    FirstName IN VARCHAR2,
    LastName IN VARCHAR2,
    City IN VARCHAR2,
    SubCity IN VARCHAR2,
    Kebele IN NUMBER,
    Salary IN NUMBER,
    Position IN VARCHAR2,
    Email IN VARCHAR2,
    Phone IN VARCHAR2,
    HireDate IN DATE
) AS
```

```
BEGIN
    INSERT INTO Employees (FirstName, LastName, City, SubCity, Kebele, Salary, Position, Email,
Phone, HireDate)
    VALUES (FirstName, LastName, City, SubCity, Kebele, Salary, Position, Email, Phone,
HireDate);
END;
/


-- Procedure to update product stock
CREATE OR REPLACE PROCEDURE UpdateProductStock(
    productId IN NUMBER,
    quantity IN NUMBER
) AS
BEGIN
    UPDATE Inventory
    SET QuantityInStock = QuantityInStock + quantity
    WHERE ProductID = productId;
    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'No product found with the given ProductID.');
    END IF;
END;
/


-- Procedure to delete an employee with dependency check
CREATE OR REPLACE PROCEDURE DeleteEmployee(
    empId IN NUMBER
) AS
    salesCount NUMBER;
BEGIN
    SELECT COUNT(*) INTO salesCount FROM Sales WHERE EmployeeID = empId;
    IF salesCount > 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Cannot delete employee with existing sales
records.');
```

```
    ELSE
        DELETE FROM Employees WHERE EmployeeID = empId;
    END IF;
END;
/

-- DDL: Create Trigger

-- Trigger to update product stock when a sale is made
CREATE OR REPLACE TRIGGER trg_UpdateProductStock
AFTER INSERT ON SaleDetails
FOR EACH ROW
BEGIN
    UPDATE Inventory
    SET QuantityInStock = QuantityInStock - :new.Quantity
    WHERE ProductID = :new.ProductID;
END;
/

-- DDL: function

-- Function to calculate total price for a product order
CREATE OR REPLACE FUNCTION CalculateTotalPrice(
    productId IN NUMBER,
    quantity IN NUMBER
) RETURN NUMBER IS
    totalPrice NUMBER(10, 2);
BEGIN
    SELECT UnitPrice * quantity INTO totalPrice
    FROM Products
    WHERE ProductID = productId;
    RETURN totalPrice;
```

```
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
/

-- Function to check product stock
CREATE OR REPLACE FUNCTION ProductStock(
    productId IN NUMBER
) RETURN NUMBER IS
    stockQuantity NUMBER;
BEGIN
    SELECT QuantityInStock INTO stockQuantity
    FROM Inventory
    WHERE ProductID = productId;
    RETURN stockQuantity;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
/

-- Function to check if a product is expired
CREATE OR REPLACE FUNCTION CheckProductExpiry(
    productId IN NUMBER
) RETURN NUMBER IS
    expiryDate DATE;
BEGIN
    SELECT ExpiryDate INTO expiryDate
    FROM Products
    WHERE ProductID = productId;
    IF expiryDate < SYSDATE THEN
```

```
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;
END;
/

-- Create View for Employees Excluding Salary
CREATE VIEW Employees_Public AS
SELECT EmployeeID, FirstName, LastName, City, SubCity, Kebele, Position, Email, Phone,
HireDate
FROM Employees;

-- DQL:

-- SELECT with specific columns: Supplier names and phone numbers
SELECT SupplierName, Phone FROM Suppliers;

-- SELECT with alias: Rename columns in output
SELECT ProductName AS Item, UnitPrice AS Price FROM Products;

-- SELECT with WHERE: Employees with salary greater than 10000
SELECT FirstName, LastName, Salary
FROM Employees
WHERE Salary > 10000;

-- SELECT with ORDER BY: Products sorted by expiry date
SELECT ProductName, ExpiryDate
FROM Products
ORDER BY ExpiryDate ASC;
```

```sql
-- SELECT with JOIN: Supplier and their order details
SELECT s.SupplierName, p.ProductName, o.OrderDate AS SupplyDay
FROM Suppliers s
JOIN Orders o ON s.SupplierID = o.SupplierID
JOIN OrderDetails od ON o.OrderID = od.OrderID
JOIN Products p ON od.ProductID = p.ProductID;

-- SELECT with multiple JOINs: Bill details with employee and customer names
SELECT b.BillID, b.BillDate, b.TotalPayment, b.PaidAmount,
    e.FirstName, e.LastName,
    c.FirstName AS CustomerFirstName, c.LastName AS CustomerLastName
FROM Bill b
JOIN Sales s ON b.SaleID = s.SaleID
JOIN Employees e ON b.EmployeeID = e.EmployeeID
JOIN Customers c ON s.CustomerID = c.CustomerID;

-- SELECT with JOIN: Store details with phone numbers
SELECT s.Name, s.Address, s.Phone
FROM Store s;

-- SELECT with GROUP BY: Number of products per category
SELECT c.CategoryName, COUNT(p.ProductID) AS NumProducts
FROM Categories c
LEFT JOIN Products p ON c.CategoryID = p.CategoryID
GROUP BY c.CategoryName;

-- SELECT with subquery: Employees who have processed sales
SELECT FirstName, LastName
FROM Employees
WHERE EmployeeID IN (SELECT DISTINCT EmployeeID FROM Sales);
```

```
-- DML: Demonstration of Data Manipulation Statements

-- SELECT with joins: Retrieve products ordered by a specific customer
SELECT c.FirstName, c.LastName, p.ProductName, sd.Quantity, sd.UnitPrice
FROM Customers c
JOIN Sales s ON c.CustomerID = s.CustomerID
JOIN SaleDetails sd ON s.SaleID = sd.SaleID
JOIN Products p ON sd.ProductID = p.ProductID
WHERE c.CustomerID = 1;

-- INSERT: Add a new product
INSERT INTO Products (ProductName, CategoryID, SupplierID, UnitPrice, Description,
ManufactureDate, ExpiryDate)
VALUES ('Tea', 2, 3, 40.00, 'GreenLeaf Tea', TO_DATE('2024-04-01', 'YYYY-MM-DD'),
TO_DATE('2026-03-31', 'YYYY-MM-DD'));

-- UPDATE: Modify a product's price
UPDATE Products
SET UnitPrice = 160.00
WHERE ProductID = 1;

-- Transaction with ROLLBACK
BEGIN
  INSERT INTO Customers (FirstName, LastName, City, SubCity, Kebele, Email, Phone)
  VALUES ('New', 'Test', 'Addis Ababa', 'Bole', 12, 'new@example.com', '+251900000000');
  -- Suppose something goes wrong, rollback the transaction
  ROLLBACK;
END;
/

-- MERGE: Update or insert supplier data
MERGE INTO Suppliers s
USING (
    SELECT 1 AS SupplierID, 'Updated AgriDistributors' AS SupplierName, 'John Doe' AS ContactName, '+251911223344' AS Phone, 'john@agridist.com' AS Email, 'Addis Ababa, Bole' AS Address FROM dual

    UNION ALL

    SELECT 6, 'New Supplier', 'Anna Lee', '+251966778899', 'anna@newsup.com', 'Dire Dawa, Gurgura' FROM dual

) t

ON (s.SupplierID = t.SupplierID)

WHEN MATCHED THEN

    UPDATE SET

        s.SupplierName = t.SupplierName,

        s.ContactName = t.ContactName,

        s.Phone = t.Phone,

        s.Email = t.Email,

        s.Address = t.Address

WHEN NOT MATCHED THEN

    INSERT (SupplierID, SupplierName, ContactName, Phone, Email, Address)

    VALUES (t.SupplierID, t.SupplierName, t.ContactName, t.Phone, t.Email, t.Address);


-- Test Functions and Procedures

SELECT CalculateTotalPrice(1, 10) AS TotalPrice FROM dual;

SELECT CheckProductExpiry(2) AS IsExpired FROM dual;

EXEC UpdateProductStock(3, 50);

SELECT ProductStock(4) AS StockQuantity FROM dual;

EXEC AddEmployee('Misrak', 'Debebe', 'Addis Ababa', 'Kolfe Keraniyo', 12, 3500.00, 'Salesperson', 'misrak@gmail.com', '+251944556677', TO_DATE('2023-04-01', 'YYYY-MM-DD'));

EXEC DeleteEmployee(6);


-- DCL: Create Roles and Assign Permissions

CREATE ROLE C##admin_role;

CREATE ROLE C##manager_role;

CREATE ROLE C##cashier_role;

CREATE ROLE C##customer_role;
```

```
-- Grant permissions to roles
GRANT SELECT, INSERT, UPDATE, DELETE ON Products TO C##admin_role;
GRANT SELECT ON Products TO C##manager_role;
GRANT SELECT ON Products TO C##cashier_role;
GRANT SELECT ON Products TO C##customer_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Bill TO C##admin_role;
GRANT SELECT, INSERT ON Bill TO C##cashier_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Employees TO C##admin_role;
GRANT SELECT ON Employees TO C##manager_role;
GRANT SELECT ON Employees TO C##cashier_role;

-- Create users and assign roles
CREATE USER C##admin_user IDENTIFIED BY password;
GRANT C##admin_role TO C##admin_user;

CREATE USER C##manager_user IDENTIFIED BY password;
GRANT C##manager_role TO C##manager_user;

CREATE USER C##cashier_user IDENTIFIED BY password;
GRANT C##cashier_role TO C##cashier_user;

CREATE USER C##customer_user IDENTIFIED BY password;
GRANT C##customer_role TO C##customer_user;

-- DQL: Check Database Status
SELECT * FROM v$version WHERE banner LIKE 'Oracle%';
```