```
-- create the Employee table if it doesn't exist
CREATE TABLE IF NOT EXISTS Employee (
    EmpID INT PRIMARY KEY,
    EmpName VARCHAR(100),
    Salary DECIMAL(10,2),
    Department VARCHAR(50)
);

-- Create Employee_Log table for logging new employees
CREATE TABLE IF NOT EXISTS Employee_Log (
    LogID INT PRIMARY KEY AUTO_INCREMENT,
    EmpID INT,
    Action VARCHAR(50),
    LogMessage VARCHAR(255),
    LogDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Create Deleted_Employees table for storing deleted employee records
CREATE TABLE IF NOT EXISTS Deleted_Employees (
    EmpID INT,
    EmpName VARCHAR(100),
    Salary DECIMAL(10,2),
    Department VARCHAR(50),
    DeletedDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Drop triggers if they exist (to avoid conflicts)
DROP TRIGGER IF EXISTS before_employee_insert;
DROP TRIGGER IF EXISTS after_employee_insert;
DROP TRIGGER IF EXISTS before_employee_update;
DROP TRIGGER IF EXISTS after_employee_delete;

-- 1. BEFORE INSERT trigger to ensure minimum salary
DELIMITER //
CREATE TRIGGER before_employee_insert
```

```
BEFORE INSERT ON Employee
FOR EACH ROW
BEGIN
   IF NEW.Salary < 3000 THEN
      SET NEW.Salary = 3000;
   END IF;
END//
DELIMITER ;

-- 2. AFTER INSERT trigger to log new employees
DELIMITER //
CREATE TRIGGER after_employee_insert
AFTER INSERT ON Employee
FOR EACH ROW
BEGIN
   INSERT INTO Employee_Log (EmpID, Action, LogMessage)
   VALUES (NEW.EmpID, 'INSERT', CONCAT('New employee added: ',
NEW.EmpName));
END//
DELIMITER ;

-- 3. BEFORE UPDATE trigger to prevent salary reduction
DELIMITER //
CREATE TRIGGER before_employee_update
BEFORE UPDATE ON Employee
FOR EACH ROW
BEGIN
   IF NEW.Salary < OLD.Salary THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Cannot reduce employee salary below
current salary';
   END IF;
END//
DELIMITER ;
```

```
-- 4. AFTER DELETE trigger to store deleted employees
DELIMITER //
CREATE TRIGGER after_employee_delete
AFTER DELETE ON Employee
FOR EACH ROW
BEGIN
    INSERT INTO Deleted_Employees (EmpID, EmpName, Salary,
Department)
    VALUES (OLD.EmpID, OLD.EmpName, OLD.Salary, OLD.Department);
END//
DELIMITER ;



-- Clear any existing data
DELETE FROM Employee;
DELETE FROM Employee_Log;
DELETE FROM Deleted_Employees;

-- Test 1: BEFORE INSERT trigger - salary adjustment
INSERT INTO Employee (EmpID, EmpName, Salary, Department)
VALUES (1, 'John Doe', 2500, 'IT');
-- Salary will be automatically increased to 3000

INSERT INTO Employee (EmpID, EmpName, Salary, Department)
VALUES (2, 'Jane Smith', 5000, 'HR');
-- Salary remains 5000

-- Check what's in the Employee table
SELECT * FROM Employee;

-- Check Employee_Log table (should have 2 entries)
SELECT * FROM Employee_Log;

-- Test 2: BEFORE UPDATE trigger - salary changes
-- This should work (increasing salary)
```

```
UPDATE Employee SET Salary = 5500 WHERE EmpID = 2;

-- This should FAIL (decreasing salary) - this is expected to error
UPDATE Employee SET Salary = 4000 WHERE EmpID = 2;

-- Test a successful update that doesn't change salary
UPDATE Employee SET Department = 'Finance' WHERE EmpID = 2;

-- Test 3: AFTER DELETE trigger
DELETE FROM Employee WHERE EmpID = 1;

-- Check Deleted_Employees table (should have John Doe's record)
SELECT * FROM Deleted_Employees;

-- Final check of all tables
SELECT 'Employee Table:' AS '';
SELECT * FROM Employee;

SELECT 'Employee_Log Table:' AS '';
SELECT * FROM Employee_Log;

SELECT 'Deleted_Employees Table:' AS '';
SELECT * FROM Deleted_Employees;
```