Trigger

```
-- Create the main table
CREATE TABLE employees (
    emp_id INT AUTO_INCREMENT PRIMARY KEY,
    emp_name VARCHAR(100),
    emp_position VARCHAR(50),
    salary DECIMAL(10, 2),
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

-- Create the audit_log table
CREATE TABLE audit_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    emp_id INT,
    action_type VARCHAR(20),
    action_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    old_salary DECIMAL(10, 2),
    new_salary DECIMAL(10, 2),
    FOREIGN KEY (emp_id) REFERENCES employees(emp_id)
);

-- AFTER INSERT Trigger
DELIMITER //

CREATE TRIGGER after_employee_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
    INSERT INTO audit_log (emp_id, action_type, new_salary)
    VALUES (NEW.emp_id, 'INSERT', NEW.salary);
END; //
```

```
DELIMITER ;

-- AFTER UPDATE Trigger
DELIMITER //

CREATE TRIGGER after_employee_update
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
   IF OLD.salary != NEW.salary THEN
      INSERT INTO audit_log (emp_id, action_type, old_salary, new_salary)
      VALUES (NEW.emp_id, 'UPDATE', OLD.salary, NEW.salary);
   END IF;
END; //

DELIMITER ;

-- test 1st trigger :Insert New Employee
INSERT INTO employees (emp_name, emp_position, salary)
VALUES ('Alice Johnson', 'Manager', 75000);

-- Check the audit_log
SELECT * FROM audit_log;

-- test 2nd trigger: Update Employee Salary
UPDATE employees
SET salary = 80000
WHERE emp_id = 1;
-- Check the audit_log
SELECT * FROM audit_log;
```

Cursor

```
CREATE TABLE employees ( emp_id INT AUTO_INCREMENT PRIMARY KEY,
    emp_name VARCHAR(100), emp_position VARCHAR(50), salary DECIMAL(10, 2));


INSERT INTO employees (emp_name, emp_position, salary)
VALUES ('Alice Johnson', 'Manager', 75000), ('Bob Smith', 'Developer', 60000), ('Charlie Brown', 'Designer', 50000);


--Using Cursor Over employees Table
DELIMITER //


-- Create a stored procedure to use a cursor
CREATE PROCEDURE simple_cursor_example()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE v_emp_id INT;
    DECLARE v_emp_name VARCHAR(100);
    DECLARE v_salary DECIMAL(10, 2);

    -- Declare the cursor
    DECLARE emp_cursor CURSOR FOR
        SELECT emp_id, emp_name, salary
        FROM employees;

    -- Declare handler for when no more rows are found
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    -- Open the cursor
    OPEN emp_cursor;

    -- Fetch each row and process it
```

```
    read_loop: LOOP
        FETCH emp_cursor INTO v_emp_id, v_emp_name, v_salary;

        -- Exit the loop when done
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- Display employee info (as an example of processing)
        SELECT v_emp_id AS Emp_ID, v_emp_name AS Emp_Name, v_salary AS Salary;

    END LOOP;

    -- Close the cursor
    CLOSE emp_cursor;
END; //

DELIMITER ;

-- Call the procedure to run the cursor
CALL simple_cursor_example();
```