**Business Case Scenario**

**Scenario**: We are developing a simple employee management system for a company. The system will track employee information, including their salary. To ensure that salary changes are logged for auditing purposes, we will implement the following:

1. A table to store employee information.
2. A function to update an employee's salary.
3. A trigger that logs the changes to another table whenever a salary update occurs.

**Step 1: Create the Tables**

**DDL for employees Table**

sql

Copy code

CREATE TABLE employees (

employee\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

department VARCHAR(50) NOT NULL,

salary NUMERIC NOT NULL CHECK (salary >= 0)

);

**DDL for salary\_changes Table (to log salary changes)**

sql

Copy code

CREATE TABLE salary\_changes (

change\_id SERIAL PRIMARY KEY,

employee\_id INT NOT NULL,

old\_salary NUMERIC NOT NULL,

new\_salary NUMERIC NOT NULL,

change\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

**Step 2: Insert Sample Data**

sql

Copy code

-- Insert sample data into employees

INSERT INTO employees (name, department, salary) VALUES

('Alice Johnson', 'HR', 50000),

('Bob Smith', 'Finance', 60000),

('Charlie Brown', 'Engineering', 70000);

**Step 3: Create a Function to Update Salary**

sql

Copy code

CREATE OR REPLACE FUNCTION update\_salary(emp\_id INT, new\_salary NUMERIC)

RETURNS VOID AS $$

DECLARE

old\_salary NUMERIC;

BEGIN

-- Get the current salary of the employee

SELECT salary INTO old\_salary FROM employees WHERE employee\_id = emp\_id;

-- Update the employee's salary

UPDATE employees SET salary = new\_salary WHERE employee\_id = emp\_id;

-- Log the salary change in the salary\_changes table

INSERT INTO salary\_changes (employee\_id, old\_salary, new\_salary)

VALUES (emp\_id, old\_salary, new\_salary);

END;

$$ LANGUAGE plpgsql;

**Step 4: Create a Trigger**

**Trigger Function**

sql

Copy code

CREATE OR REPLACE FUNCTION log\_salary\_change()

RETURNS TRIGGER AS $$

BEGIN

-- Log the salary change in the salary\_changes table

INSERT INTO salary\_changes (employee\_id, old\_salary, new\_salary)

VALUES (NEW.employee\_id, OLD.salary, NEW.salary);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

**Trigger Definition**

sql

Copy code

CREATE TRIGGER salary\_change\_trigger

AFTER UPDATE OF salary ON employees

FOR EACH ROW

WHEN (OLD.salary IS DISTINCT FROM NEW.salary) -- Only log if the salary changes

EXECUTE FUNCTION log\_salary\_change();

**Step 5: Update Salary Using the Function**

Now you can update an employee's salary using the created function:

sql

Copy code

-- Update Bob Smith's salary

SELECT update\_salary(2, 65000); -- Update salary from 60000 to 65000

**Step 6: Querying the Tables**

**View Employees**

sql

Copy code

SELECT \* FROM employees;

**View Salary Changes Log**

sql

Copy code

SELECT \* FROM salary\_changes;

**Explanation**

1. **Tables**:
   * employees: Stores employee information.
   * salary\_changes: Logs salary changes for auditing purposes.
2. **Function (update\_salary)**:
   * Takes an employee ID and a new salary as input.
   * Updates the salary of the specified employee and logs the change in the salary\_changes table.
3. **Trigger (salary\_change\_trigger)**:
   * Fires after an update on the salary column of the employees table.
   * Calls the log\_salary\_change function to log the old and new salary.

**Sample Output After Salary Update**

After running the salary update query, you can check the employees and salary\_changes tables to see the changes.

* **Employees Table**:

| **employee\_id** | **name** | **department** | **salary** |
| --- | --- | --- | --- |
| 1 | Alice Johnson | HR | 50000 |
| 2 | Bob Smith | Finance | 65000 |
| 3 | Charlie Brown | Engineering | 70000 |

* **Salary Changes Table**:

| **change\_id** | **employee\_id** | **old\_salary** | **new\_salary** | **change\_date** |
| --- | --- | --- | --- | --- |
| 1 | 2 | 60000 | 65000 | 2024-10-05 12:00:00 |

**Summary**

This example demonstrates how to create a simple employee management system in PostgreSQL that uses tables, functions, PL/pgSQL, and triggers to manage and log salary changes. You can extend this scenario further by adding more features as needed!