Create a Database: Set up a database from your interns id.
Create Tables: Define two tables:

- `Departments` with `DepartmentId` and `DepartmentName`.
- `Employees` with `EmployeeId`, `FullName`, `DepartmentId`, `Salary`, and `HireDate`.

Insert Sample Data: Populate the tables with sample values:

- Departments: **HR**, **Finance**, **Engineering**.
- Employees:
    - Alice Johnson (HR, $50,000).
    - Bob Smith (Finance, $60,000).
    - Carol Lee (Engineering, $75,000).

Write a Stored Procedure: Create a procedure `ManageEmployees` to:

- Add a new employee.
- Update an employee's salary.
- Retrieve all employees in a specific department.

Execute and Test: Use the procedure for the following:

- Add **David Miller** (Finance, $65,000).
- Update **Bob Smith**'s salary to **$70,000**.
- Retrieve all employees from the **Engineering** department.

```
CREATE TABLE Departments (
    DepartmentId SERIAL PRIMARY KEY,
    DepartmentName VARCHAR(50) NOT NULL
);
CREATE TABLE Employees (
    EmployeeId SERIAL PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    DepartmentId INT NOT NULL,
    Salary NUMERIC(10, 2),
    HireDate DATE DEFAULT CURRENT_DATE,
    FOREIGN KEY (DepartmentId) REFERENCES Departments(DepartmentId)
);
INSERT INTO Departments (DepartmentName)
VALUES ('HR'), ('Finance'), ('Engineering');

INSERT INTO Employees (FullName, DepartmentId, Salary)
VALUES
('Alice Johnson', 1, 50000),
('Bob Smith', 2, 60000),
('Carol Lee', 3, 75000);
```

```
CREATE OR REPLACE PROCEDURE ManageEmployees(
    op_type VARCHAR,
    emp_name VARCHAR DEFAULT NULL,
    dept_id INT DEFAULT NULL,
    emp_salary NUMERIC DEFAULT NULL
)
LANGUAGE plpgsql
AS $$
DECLARE
    emp_row RECORD; -- To store each row retrieved
BEGIN
    IF op_type = 'Add' THEN
        INSERT INTO Employees (FullName, DepartmentId, Salary)
        VALUES (emp_name, dept_id, emp_salary);

    ELSIF op_type = 'Update' THEN
        UPDATE Employees
        SET Salary = emp_salary
        WHERE FullName = emp_name;
```

```
    ELSIF op_type = 'Retrieve' THEN
        -- Loop through each row and print it

        FOR emp_row IN SELECT * FROM Employees WHERE DepartmentId = dept_id LOOP
            RAISE NOTICE 'EmployeeId: %, FullName: %, DepartmentId: %, Salary: %',
                emp_row.EmployeeId, emp_row.FullName, emp_row.DepartmentId, emp_row.Salary;
        END LOOP;
    ELSE
        RAISE NOTICE 'Invalid operation type';
    END IF;
END;
$$;
```

Add a new employee:

```
CALL ManageEmployees('Add', 'David Miller', 2, 65000);
```

Update an employee's salary:

```
CALL ManageEmployees('Update', 'Bob Smith', NULL, 70000);
```

Retrieve all employees from the Engineering department:

```
CALL ManageEmployees('Retrieve', NULL, 3, NULL);
```