Project :-

Employee Management Portal

🎯 2. Objective

This project aims to create a real-world Employee Management System using PostgreSQL that manages employees, their attendance, salary records, and hierarchical reporting. The system uses advanced SQL features like triggers, functions, constraints, and relational design.

🧱 3. Database Structure

The database contains the following main tables:

1. employees – stores personal and job-related info

2. departments – contains department names and locations

3. jobs – defines job titles and roles

4. attendance – tracks employee daily presence

5. salary_history – logs salary changes with triggers

6. leave_requests – manages leave applications

Relationships:

- One department has many employees

- One employee can be a manager of others (self-reference)

- One employee can have many attendance and leave records

⚙️ 4. Key Features

- Self-referencing foreign key for manager system

- Triggers to track salary changes

- Prevent employee from being their own manager

- Function to calculate present days

- Normalized relational design

- Sample data with 15+ entries per table

🧪 5. Functions and Triggers Used

1. Trigger: log_salary_changes – saves salary update to history

2. Trigger: prevent_self_manager – prevents assigning self as manager

3. Function: get_present_days(emp_id) – returns total present days

💾 6. Technologies Used

- PostgreSQL

- PL/pgSQL (triggers and functions)

- SQL constraints (foreign key, primary key, not null, unique)

📌 7. Conclusion

The project successfully demonstrates practical use of PostgreSQL for a real-life HR system, including employee hierarchy, attendance management, and salary auditing.

Here are my syntax which i used in my project

Tables

```
create table departments(
    dept_id serial primary key,
       dept_name varchar(100)
);


create table jobs(
   job_id serial primary key,
```

```
    job_title text not null,
    min_salary numeric not null,
    max_salary numeric not null
);

create table employees(
  emp_id serial primary key,
    first_name text not null,
    last_name text not null,
    email text unique not null,
    phone text,
    hire_date date not null,
    job_id int references jobs(job_id),
    salary numeric not null,
    dept_id int references department(dept_id),
    manager_id int references employees(emp_id)
);

create table attendance(
   att_id serial primary key,
     emp_id int references employees(emp_id),
     attendance_date date not null,
     status text check (status in ('Present', 'Absent', 'Leave')) not null
);

create table salary_history(
```

record_id serial primary key,

emp_id int references employees(emp_id),

change_date timestamp default now(),

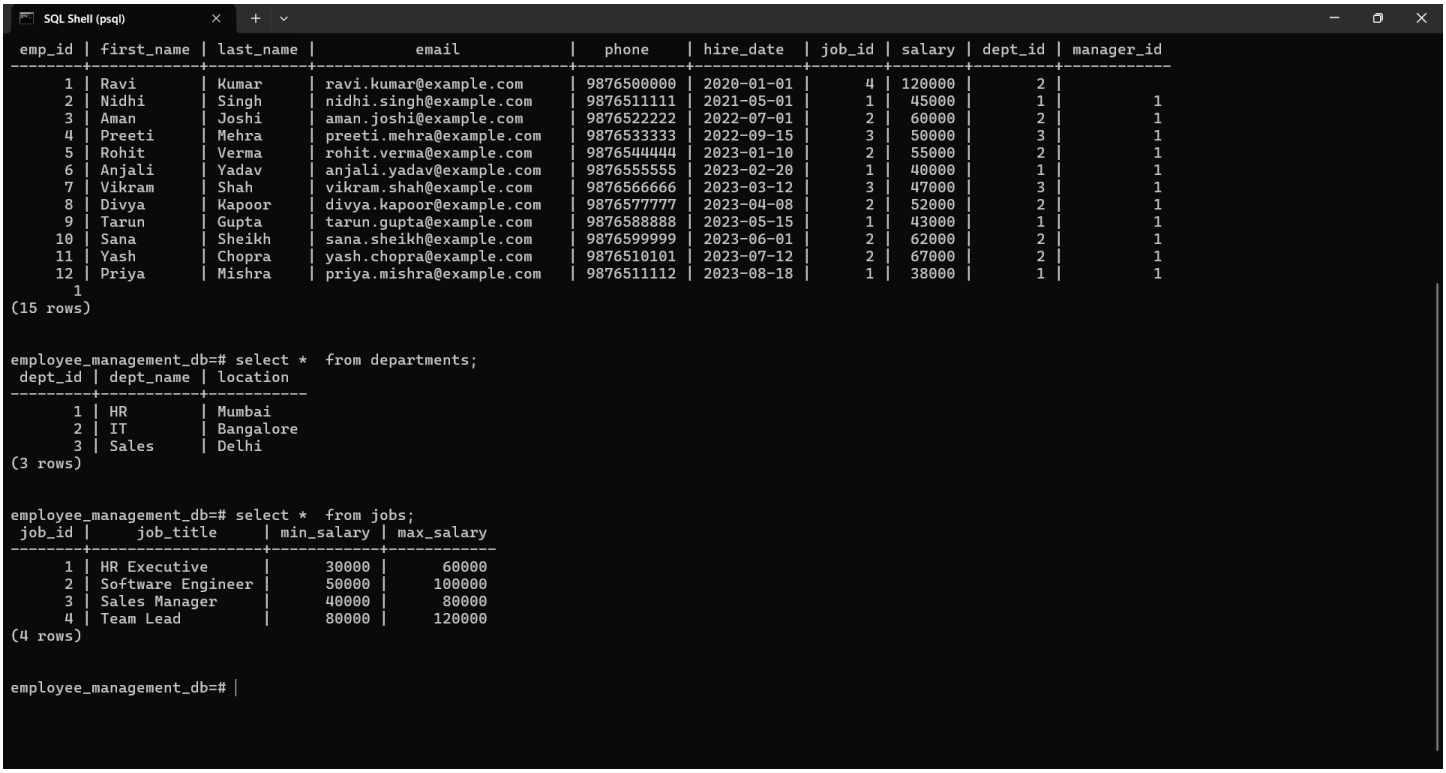
old_salary numeric,

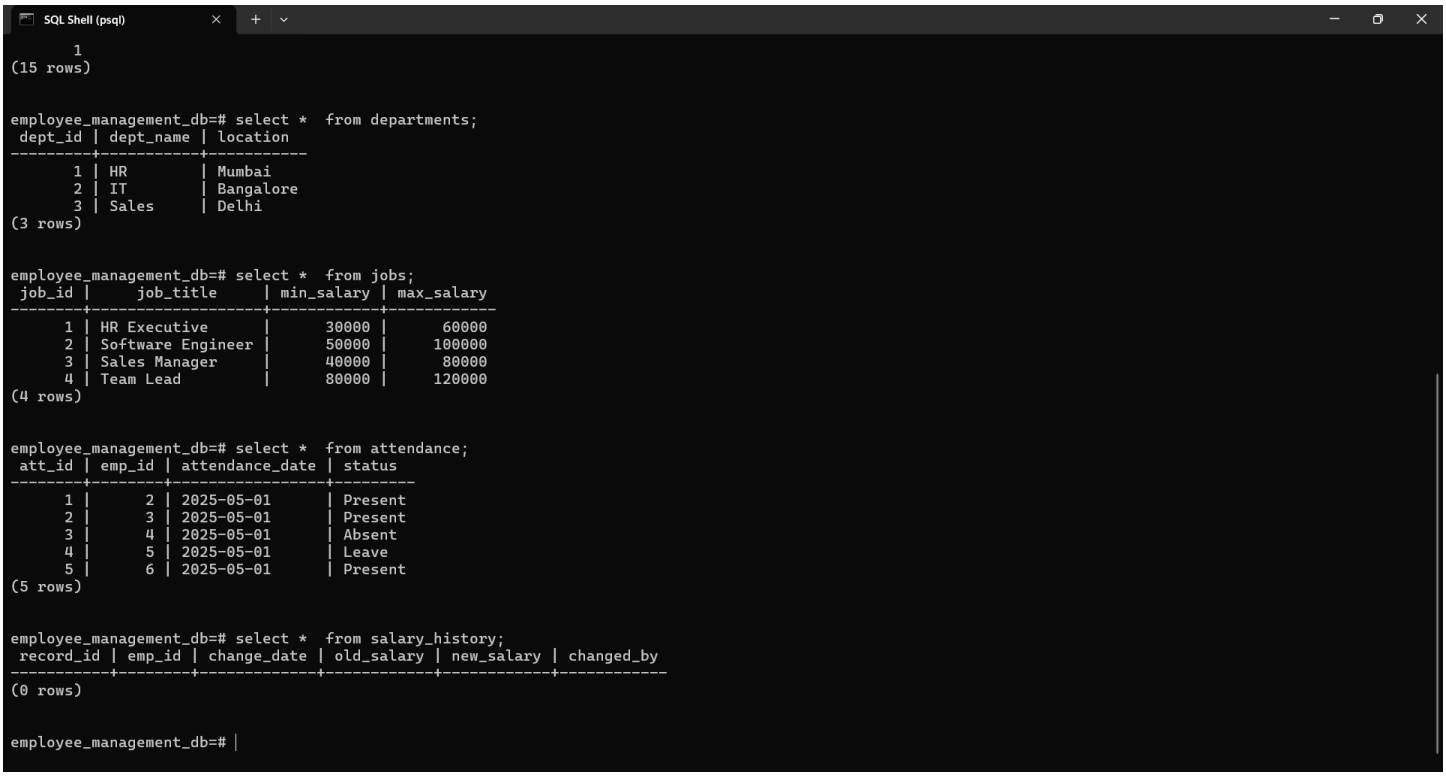
new_salary numeric,

changed_by text

);

Data which i insert in this tables:-





Functions which i made :-

-- Trigger Function

CREATE OR REPLACE FUNCTION log_salary_changes()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.salary <> OLD.salary THEN

INSERT INTO salary_history (emp_id, old_salary, new_salary, changed_by)

VALUES (OLD.emp_id, OLD.salary, NEW.salary, current_user);

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

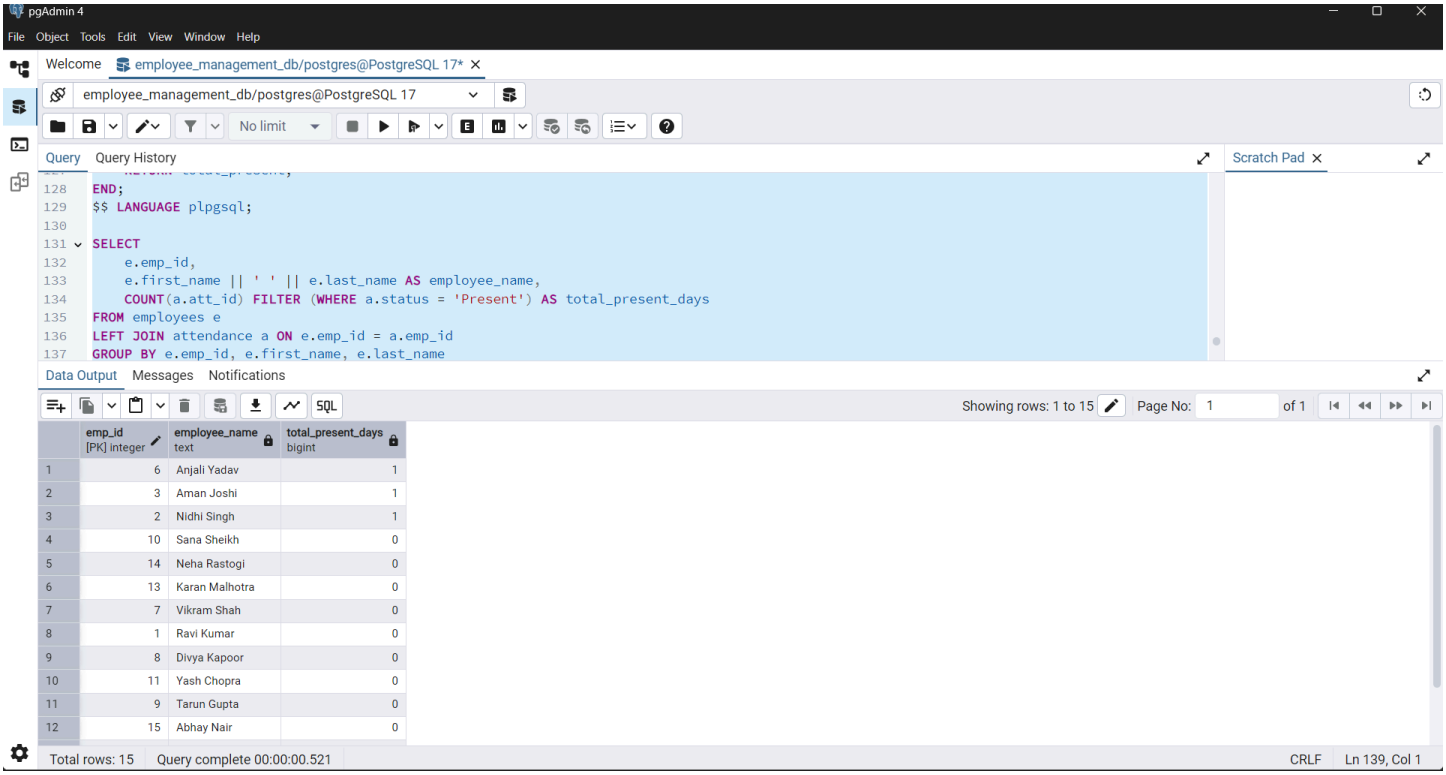
-- Trigger on employees table

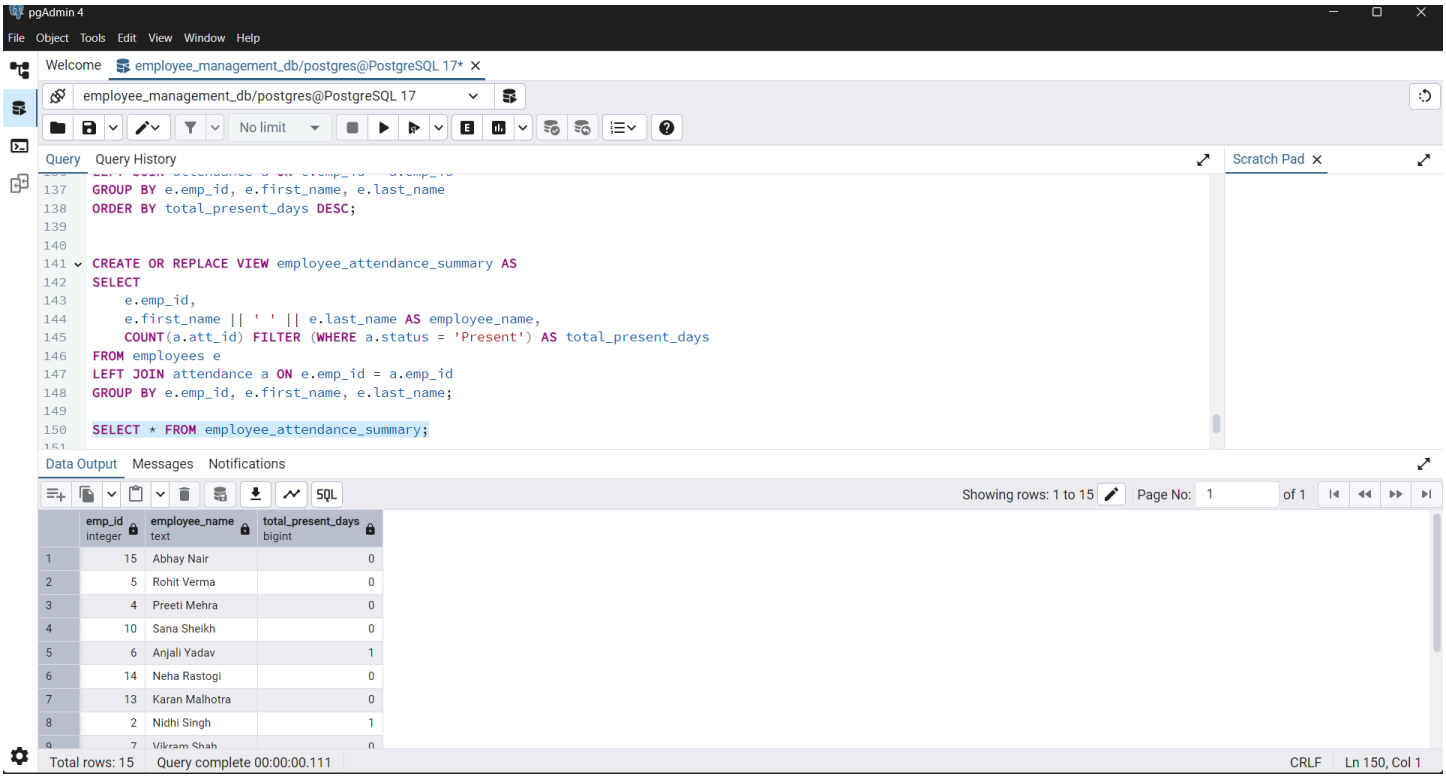
```
CREATE TRIGGER trg_salary_change
AFTER UPDATE ON employees
FOR EACH ROW
WHEN (OLD.salary IS DISTINCT FROM NEW.salary)
EXECUTE FUNCTION log_salary_changes();

-- Trigger Function
CREATE OR REPLACE FUNCTION prevent_self_manager()
RETURNS TRIGGER AS $$
BEGIN
  IF NEW.manager_id = NEW.emp_id THEN
    RAISE EXCEPTION 'Employee cannot be their own manager';
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_prevent_self_manager
BEFORE INSERT OR UPDATE ON employees
FOR EACH ROW
EXECUTE FUNCTION prevent_self_manager();

 --Function: Calculate total present days of an employee
CREATE OR REPLACE FUNCTION get_present_days(p_emp_id INTEGER)
RETURNS INTEGER AS $$
DECLARE
```

```
    total_present INTEGER;
BEGIN
    SELECT COUNT(*) INTO total_present
    FROM attendance
    WHERE emp_id = p_emp_id AND status = 'Present';

    RETURN total_present;
END;
$$ LANGUAGE plpgsql;

SELECT
    e.emp_id,
    e.first_name || ' ' || e.last_name AS employee_name,
    COUNT(a.att_id) FILTER (WHERE a.status = 'Present') AS total_present_days
FROM employees e
LEFT JOIN attendance a ON e.emp_id = a.emp_id
GROUP BY e.emp_id, e.first_name, e.last_name
ORDER BY total_present_days DESC;

CREATE OR REPLACE VIEW employee_attendance_summary AS
SELECT
    e.emp_id,
    e.first_name || ' ' || e.last_name AS employee_name,
    COUNT(a.att_id) FILTER (WHERE a.status = 'Present') AS total_present_days
FROM employees e
```

LEFT JOIN attendance a ON e.emp_id = a.emp_id

GROUP BY e.emp_id, e.first_name, e.last_name;





Thank you