📄 IMPLEMENTATION FILES – DATABASE SCHEMA DESIGN OBJECTIVE

◆ Overview:

This section documents the file(s) that contain the core logic and functional implementation of the database schema design for the Employee Directory System. The schema is built to support a scalable, multi-company environment with well-normalized tables and enforced relationships.

The implementation is structured into SQL scripts and divided logically into:

- One script for static mappings and lookups (e.g., roles, departments, designations)
- One script for core data structures (e.g., employees, projects, tasks)

◆ Objective:

Create database schemas for:

- **Employees**: Store employee name, role, department, email, phone number, profile picture URL, and bio.
- **Departments**: Store available department names.

Output:
A `schema.sql` script defining the database schema.

◆ Directory Structure:

```
backend/
├── schema.sql
└── schema_mapping.sql
```

◆ File Descriptions:

📄 `schema.sql`

- Contains the creation scripts for all core **employee-related** tables.
- Includes:
    - `employees`
    - `projects`
    - `certifications`
    - `experience`
    - `skills`
    - `tasks`

- Defines full structure with:
  - Primary keys and foreign keys
  - Data validation constraints (CHECK, NOT NULL, UNIQUE)
  - Relations to mapping tables (`roles`, `departments`, etc.)
- Stores core data such as:
  - Name, email, contact info, gender
  - Department and role references
  - Bio, profile image, skills, experience summary
  - Performance statistics and task overview

`schema_mapping.sql`

- Contains static reference tables and dropdown options used in `employees`.
- Includes:
  - `companies`
  - `designations`
  - `main_departments`
  - `sub_departments`
  - `roles`
  - `role_department_mapping`
- Used to enforce controlled values and valid role-department-designation combinations.
- Prevents manual entry errors and ensures integrity of role assignments in HR workflows.

◆ Modular Implementation Style:

- Core schema separated from lookup and mapping logic.
- Enables flexibility for updating mappings without affecting employee data.
- Supports HR-controlled dropdowns using backend-linked enums.
- Highly normalized structure (3NF compliance).

◆ Compliance:

- Implements referential integrity using `FOREIGN KEY` constraints.
- Optimized for real-time search, filter, and relationship-based queries.
- Aligned with scalable SaaS directory systems.
- All data types and constraints selected for clarity, performance, and validation safety.

Let me know if you want this styled for PDF or combined with your other documents!