1. Employees

```
CREATE TABLE Employees(
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    password VARCHAR(100),
    role ENUM('TEAM_LEADER', 'PROJECT_MANAGER', 'HR', 'INTERVIEWER',
'BENCH'),
    salary DECIMAL(10,2) DEFAULT 0.00,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2. Job_requests

```
CREATE TABLE job_requests (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(255),
    description TEXT,
    required_employees INT,
    created_by INT,                                    -- TL
    status ENUM('PENDING_PM', 'FORWARDED_HR', 'CLOSED') DEFAULT
'PENDING_PM',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (created_by) REFERENCES Employees(id)
);
```

Step 1: Job Request by Team Leader (TL)

- The **Team Leader** logs in and creates a **Job Request** by filling:
    - Job Title
    - Job Description
    - Number of required employees

- This data is saved in the job_requests table.
- The status of the request is set to 'PENDING_PM'.

3. Project_budgets

```
CREATE TABLE project_budgets (
    id INT PRIMARY KEY AUTO_INCREMENT,
    job_request_id INT,
    pm_id INT,                    -- PM who analyzed and created budget
    total_budget DECIMAL(12,2),
    bench_salary_total DECIMAL(12,2),
    calculated_ctc_per_candidate DECIMAL(12,2),
    calculated_new_hires INT,
    forwarded_to_hr BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (job_request_id) REFERENCES job_requests(id),
    FOREIGN KEY (pm_id) REFERENCES Employees(id)
);
```

4. Bench_assignments

```
CREATE TABLE bench_assignments (
    id INT PRIMARY KEY AUTO_INCREMENT,
    employee_id INT,
    job_request_id INT,
    FOREIGN KEY (employee_id) REFERENCES employee(id),
    FOREIGN KEY (job_request_id) REFERENCES job_requests(id)
);
```

step 2: PM Analyzes the Request

- The **Project Manager** logs in and views all job requests with status = 'PENDING_PM'.
- PM checks if there are **bench employees** available.

a. If Bench Employees Are Available:

- PM assigns them to the request (bench_assignments table).
- Calculates total salary of bench employees from the **employee**.salary field.
- Reduces this from the total **project budget**.

b. If Not Enough Bench Employees:

- PM decides a **project budget** and inputs it.
- Calculates **CTC per new candidate**:

total_budget - bench_salary_total = remaining_budget
ctc_per_candidate = remaining_budget / number_of_new_candidates

This data is saved in the project_budgets table with:

- Total budget
- Bench salary total
- CTC per new employee
- Number of new employees needed
- Sets forwarded_to_hr = TRUE

Updates job_requests.status to 'FORWARDED_HR'

5. new_candidates

```
CREATE TABLE new_candidates (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(15),
    resume_link TEXT,
    skills TEXT,
    added_by INT,
    status ENUM('PENDING_REVIEW', 'APPROVED', 'REJECTED') DEFAULT
'PENDING_REVIEW',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (added_by) REFERENCES employee(id)
);
```

Step 3: HR Processes the Request

- HR logs in and views requests where:
    - status = 'FORWARDED_HR'
    - And project_budgets.forwarded_to_hr = TRUE
- HR views:
    - Project budget
    - CTC per candidate
    - Required number of new hires

a. Posting & Candidate Management:

- HR posts the job publicly (external to system).

- HR adds new_candidates to the system.
- After reviewing, selected ones are promoted to candidates table.

6. Candidates

```
CREATE TABLE candidates (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(15),
    resume_link TEXT,
    skills TEXT,
    added_by INT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (added_by) REFERENCES employee(id)
);
```

7. Applications

```
CREATE TABLE applications (
    id INT PRIMARY KEY AUTO_INCREMENT,
    candidate_id INT,
    job_request_id INT,
    ctc_offered DECIMAL(10,2),
```

```
    status ENUM('PENDING', 'INTERVIEWED', 'SELECTED', 'REJECTED',
'CONFIRMED'),

    applied_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (candidate_id) REFERENCES candidates(id),

    FOREIGN KEY (job_request_id) REFERENCES job_requests(id)

);
```

Step 4: Candidate Applications

- HR maps candidates to a job request via the applications table.
- For each application:
    - HR sets the ctc_offered (usually equal to the calculated CTC).
    - Status is marked as 'PENDING'.

8. interviews

```
CREATE TABLE interviews (

    id INT PRIMARY KEY AUTO_INCREMENT,

    application_id INT,

    interviewer_id INT,

    result ENUM('SELECTED', 'REJECTED'),

    feedback TEXT,

    interview_date TIMESTAMP,

    FOREIGN KEY (application_id) REFERENCES applications(id),
```

```
    FOREIGN KEY (interviewer_id) REFERENCES employee(id)
);
```

Step 5: Interview Process

- The **Interviewer** logs in and views applications assigned.
- Conducts interview and records:
    - Result (SELECTED or REJECTED)
    - Feedback
- Stored in interviews table.

Step 6: Finalization by HR

- If selected:
    - HR confirms and updates applications.status = 'CONFIRMED'
    - Sends a confirmation mail and document checklist to the candidate.
- If not selected:
    - Status is set to 'REJECTED'

Step 7: Closing the Request

- If the number of confirmed candidates + bench employees = required employees:
    - HR or PM closes the job request:
        - job_requests.status = 'CLOSED'

Action	Table Involved
TL creates job	job_requests
PM assigns bench & budget	bench_assignments, project_budgets
HR adds/view candidates	new_candidates, candidates
HR maps candidate to job	applications
Interviewer evaluates	interviews
HR finalizes offer	applications, mail system