Table of Contents

📖 Overview

This system handles employee shift management, including:

- Custom ID generation
- Availability and leave tracking
- Shift assignment (manual and automatic)
- Shift swap requests
- Auditing and reporting

All operations ensure data consistency through foreign key constraints, triggers, and stored procedures.

Schema Design

1. id_counters

Manages sequential ID generation for different entities.

SQL query:

```
CREATE TABLE id_counters (
  id_type VARCHAR(50) PRIMARY KEY,
  last_number INT NOT NULL
);
```

2. employees

Stores employee details. Each employee has a unique ID and role.

SQL query:

```
CREATE TABLE employees (
  employee_id VARCHAR(10) PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100),
  phone VARCHAR(10),
  role VARCHAR(50),
  status ENUM('active', 'inactive') DEFAULT 'active',
  CONSTRAINT unique_name_phone UNIQUE(name, phone)
);
```

3. shifts

Defines time slots for employee shifts.

SQL query

```
CREATE TABLE shifts (
  shift_id VARCHAR(10) PRIMARY KEY,
  shift_name VARCHAR(50),
  start_time TIME NOT NULL,
  end_time TIME NOT NULL
);
```

4. employee_availability

Tracks which employees are available on which days.

SQL query:

```
CREATE TABLE employee_availability (
    availability_id INT AUTO_INCREMENT PRIMARY KEY,
    employee_id VARCHAR(10) NOT NULL,
    available_date DATE NOT NULL,
    FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
);
```

5. leave_requests

Manages leave applications for employees.

SQL query:

```
CREATE TABLE leave_requests (
    leave_id INT AUTO_INCREMENT PRIMARY KEY,
    employee_id VARCHAR(10) NOT NULL,
    leave_start DATE,
    leave_end DATE,
    reason TEXT,
    status ENUM('pending', 'approved', 'rejected') DEFAULT 'pending',
    FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
);
```

6. shift_assignments

Tracks shift allocation. Enforces:

- No duplicate assignments per day
- Valid employees and shifts

SQL query:

```
CREATE TABLE shift_assignments (
    assignment_id INT AUTO_INCREMENT PRIMARY KEY,
    employee_id VARCHAR(10) NOT NULL,
    name VARCHAR(100) NOT NULL,
    shift_id VARCHAR(10) NOT NULL,
    shift_date DATE NOT NULL,
    assigned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (employee_id) REFERENCES employees(employee_id),
    FOREIGN KEY (shift_id) REFERENCES shifts(shift_id),
    CONSTRAINT unique_shift_per_day UNIQUE (employee_id, shift_date)
);
```

7. shift_swap_requests

Allows employees to request shift swaps.

SQL query:

```
CREATE TABLE shift_swap_requests (
    request_id INT AUTO_INCREMENT PRIMARY KEY,
    requester_id VARCHAR(10) NOT NULL,
    requester_name VARCHAR(100),
    requested_with_id VARCHAR(10) NOT NULL,
    requested_with_name VARCHAR(100),
    shift_date DATE,
    status ENUM('pending', 'approved', 'rejected') DEFAULT 'pending',
    FOREIGN KEY (requester_id) REFERENCES employees(employee_id),
    FOREIGN KEY (requested_with_id) REFERENCES employees(employee_id)
);
```

8. audit_log

Stores logs of shift assignments for traceability.

SQL query:

```
CREATE TABLE audit_log (
  log_id INT AUTO_INCREMENT PRIMARY KEY,
  employee_id VARCHAR(10),
  name VARCHAR(100),
  shift_id VARCHAR(10),
  shift_date DATE,
  action ENUM('assigned', 'updated', 'removed', 'auto-assigned'),
  changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

🧮 Stored Procedures

- **get_next_id(type, prefix, OUT id):** Generates custom IDs like EMP001.
- **add_employee(name, email, phone, role):** Creates an employee with generated ID.
- **add_shift(name, start_time, end_time):** Creates a shift with generated ID.
- **assign_shift(employee_id, name, shift_id, date):** Assigns a shift manually.
- **auto_assign_shift(shift_id, date, role):** Assigns a shift automatically to an eligible and available employee based on: **Status, Role, Availability, No conflict with approved leave, No existing shift on that day**

🔥 Trigger

trg_validate_shift_assignment

Runs before insert on shift_assignments:

- Rejects if employee is inactive
- Rejects if employee is on approved leave
- Rejects if overlapping with another shift on same day

📊 Views

- **shift_coverage:** Shows how many employees are assigned per shift.
- **employee_workload_summary:** Total shifts assigned per employee.
- **available_unassigned:** Available employees who aren't yet assigned for a given date.

🧪 Sample Data

Add Employees

CALL add_employee('Mayurima Sarkar', 'mayurima@gmail.com', '9976243102', 'cashier');

...

Add Shifts

CALL add_shift('Morning', '08:00:00', '12:00:00');

Availability

INSERT INTO employee_availability (employee_id, available_date)

VALUES ('EMP010', '2025-06-24');

Leave

INSERT INTO leave_requests (employee_id, leave_start, leave_end, reason, status)

VALUES ('EMP013', '2025-06-24', '2025-06-25', 'Personal leave', 'approved');

Assign Shifts Manually

CALL assign_shift('EMP010', 'Mayurima Sarkar', 'SFT002', '2025-06-24');

Auto Assign

CALL auto_assign_shift('SFT001', '2025-06-24', 'manager');

🧯 Common Errors & Fixes

- ❌ Cannot add or update a child row

  1.The referenced employee_id or shift_id does not exist

  2.Fix by checking employees and shifts tables

- ❌ Overlapping Shift

  1.Trigger blocks insertion if new shift overlaps with existing one

👩‍💻 Usage Guide

- Insert new employee: CALL add_employee(...)
- Set availability: INSERT INTO employee_availability
- Assign shift: CALL assign_shift(...)
- Request swap: INSERT INTO shift_swap_requests
- View reports: SELECT * FROM shift_coverage

🚀 Future Improvements

- Shift conflict visualization in UI
- Notification system for swap approvals
- Granular audit logs (who approved/rejected swaps)
- Support for partial day shifts