1. Teachers Table

```sql
CREATE TABLE Teachers (
   teacher_id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   phone VARCHAR(15),
   pass VARCHAR(255) NOT NULL
);
```

Purpose

Stores information about teachers who manage subjects and take attendance.

Columns

- **teacher_id (INT, PRIMARY KEY, AUTO_INCREMENT)**:
  - A unique identifier for each teacher, automatically generated (e.g., 1, 2, 3).
  - Used as a reference in other tables (Subjects, Attendance).
- **first_name (VARCHAR(50), NOT NULL)**:
  - Teacher's first name (e.g., "John"). Limited to 50 characters, required.
- **last_name (VARCHAR(50), NOT NULL)**:
  - Teacher's last name (e.g., "Doe"). Required field.
- **email (VARCHAR(100), UNIQUE, NOT NULL)**:
  - Teacher's email (e.g., "john.doe@example.com"). Unique to identify teachers for login; required.
- **phone (VARCHAR(15))**:
  - Teacher's phone number (e.g., "+91-1234567890"). Optional (nullable), supports various formats.
- **pass (VARCHAR(255), NOT NULL)**:
  - Teacher's password (hashed, e.g., using bcrypt). Required for authentication.

Constraints

- PRIMARY KEY (teacher_id): Ensures every teacher has a unique ID.

- UNIQUE (email): Prevents duplicate email addresses.
- NOT NULL on key fields ensures essential data is always provided.

Role in System

- Teachers log in using email and pass.
- teacher_id links to Subjects (who teaches what) and Attendance (who records it).

---

2. Students Table

```sql
CREATE TABLE Students (
   student_id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   roll_number VARCHAR(20) UNIQUE NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   phone VARCHAR(15)
);
```

Purpose

Stores details of students whose attendance is tracked.

Columns

- **student_id (INT, PRIMARY KEY, AUTO_INCREMENT)**:
  - Unique ID for each student, auto-generated.
- **first_name (VARCHAR(50), NOT NULL)**:
  - Student's first name (e.g., "Alice").
- **last_name (VARCHAR(50), NOT NULL)**:
  - Student's last name (e.g., "Smith").
- **roll_number (VARCHAR(20), UNIQUE, NOT NULL)**:
  - Unique identifier like "A001" or "2023CS001". Used for quick reference.
- **email (VARCHAR(100), UNIQUE, NOT NULL)**:

    - Student's email (e.g., "alice.smith@example.com"). Unique and required.
- **phone (VARCHAR(15))**:
    - Optional phone number (e.g., "987-654-3210").

Constraints

- PRIMARY KEY (student_id): Unique student identifier.
- UNIQUE (roll_number): Ensures no duplicate roll numbers.
- UNIQUE (email): Prevents email reuse.

Role in System

- Links to Student_Subject (enrollment) and Attendance (presence tracking).
- Teachers see students by roll_number or name when marking attendance.

---

3. Subjects Table

```
CREATE TABLE Subjects (
   subject_id INT PRIMARY KEY AUTO_INCREMENT,
   subject_name VARCHAR(100) NOT NULL,
   teacher_id INT,
   FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id) ON DELETE SET NULL
);
```

Purpose

Defines academic subjects and assigns them to teachers.

Columns

- **subject_id (INT, PRIMARY KEY, AUTO_INCREMENT)**:
    - Unique ID for each subject (e.g., 1 for "Math").
- **subject_name (VARCHAR(100), NOT NULL)**:
    - Name of the subject (e.g., "Mathematics"). Required.
- **teacher_id (INT, FOREIGN KEY)**:

    - References Teachers.teacher_id to indicate who teaches it (e.g., Teacher 1).

Constraints

- PRIMARY KEY (subject_id): Unique subject identifier.
- FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id) ON DELETE SET NULL:
    - Links to a teacher. If the teacher is deleted, teacher_id becomes NULL (subject remains unassigned).

Role in System

- Ties teachers to their subjects, ensuring they only manage attendance for their own classes.
- Links to Timetable for scheduling.

---

4. Timetable Table

```
CREATE TABLE Timetable (
   timetable_id INT PRIMARY KEY AUTO_INCREMENT,
   subject_id INT,
   day_of_week ENUM('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday') NOT NULL,
   start_time TIME NOT NULL,
   end_time TIME NOT NULL,
   room VARCHAR(50),
   semester_start_date DATE NOT NULL,
   semester_end_date DATE NOT NULL,
   UNIQUE (subject_id, day_of_week, start_time),
   CHECK (start_time < end_time),
   FOREIGN KEY (subject_id) REFERENCES Subjects(subject_id) ON DELETE CASCADE
);
```

Purpose

Defines the recurring weekly schedule for subjects.

Columns

- **timetable_id (INT, PRIMARY KEY, AUTO_INCREMENT)**:
  - Unique ID for each timetable entry.
- **subject_id (INT, FOREIGN KEY)**:
  - Links to Subjects.subject_id (e.g., Math).
- **day_of_week (ENUM, NOT NULL)**:
  - Day of the week (e.g., "Monday"). Restricted to valid days.
- **start_time (TIME, NOT NULL)**:
  - Class start time (e.g., "09:00:00").
- **end_time (TIME, NOT NULL)**:
  - Class end time (e.g., "10:00:00").
- **room (VARCHAR(50))**:
  - Optional classroom (e.g., "Room 101").
- **semester_start_date (DATE, NOT NULL)**:
  - Start of the active period (e.g., "2025-03-01").
- **semester_end_date (DATE, NOT NULL)**:
  - End of the active period (e.g., "2025-06-30").

Constraints

- UNIQUE (subject_id, day_of_week, start_time):
  - Prevents a subject from being scheduled twice at the same time on the same day.
- CHECK (start_time < end_time):
  - Ensures logical time ranges.
- FOREIGN KEY (subject_id) REFERENCES Subjects(subject_id) ON DELETE CASCADE:
  - If a subject is deleted, its timetable entries are removed.

Role in System

- Provides the recurring pattern for the calendar view (e.g., "Math every Monday, 9–10 AM").
- semester_start_date and semester_end_date define the period for session generation.

---

5. Sessions Table

```sql
CREATE TABLE Sessions (
    session_id INT PRIMARY KEY AUTO_INCREMENT,
    timetable_id INT,
    date DATE NOT NULL,
    status ENUM('Scheduled', 'Completed', 'Cancelled') DEFAULT 'Scheduled',
    UNIQUE (timetable_id, date),
    FOREIGN KEY (timetable_id) REFERENCES Timetable(timetable_id) ON DELETE CASCADE
);
```

Purpose

Tracks specific instances of classes based on the timetable.

Columns

- **session_id (INT, PRIMARY KEY, AUTO_INCREMENT)**:
  - Unique ID for each session.
- **timetable_id (INT, FOREIGN KEY)**:
  - Links to Timetable.timetable_id.
- **date (DATE, NOT NULL)**:
  - Specific date of the session (e.g., "2025-03-24").
- **status (ENUM, DEFAULT 'Scheduled')**:
  - 'Scheduled': Planned but not yet held.
  - 'Completed': Held and attendance taken.
  - 'Cancelled': Didn't happen (e.g., holiday).

Constraints

- UNIQUE (timetable_id, date):
  - Prevents duplicate sessions for the same timetable entry on the same date.
- FOREIGN KEY (timetable_id) REFERENCES Timetable(timetable_id) ON DELETE CASCADE:
  - Deletes sessions if the timetable entry is removed.

Role in System

- Populates the calendar with specific dates.
- status ensures only 'Completed' sessions count for attendance calculations.

---

6. Attendance Table

```
CREATE TABLE Attendance (
   attendance_id INT PRIMARY KEY AUTO_INCREMENT,
   student_id INT,
   session_id INT,
   status ENUM('Present', 'Absent', 'Late') DEFAULT NULL,
   timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
   recorded_by INT,
   UNIQUE (student_id, session_id),
   FOREIGN KEY (student_id) REFERENCES Students(student_id) ON DELETE CASCADE,
   FOREIGN KEY (session_id) REFERENCES Sessions(session_id) ON DELETE CASCADE,
   FOREIGN KEY (recorded_by) REFERENCES Teachers(teacher_id) ON DELETE SET NULL
);
```

Purpose

Records student attendance for each session.

Columns

- **attendance_id (INT, PRIMARY KEY, AUTO_INCREMENT):**
  - Unique ID for each attendance record.
- **student_id (INT, FOREIGN KEY):**
  - Links to Students.student_id.
- **session_id (INT, FOREIGN KEY):**
  - Links to Sessions.session_id.
- **status (ENUM, DEFAULT NULL):**
  - 'Present', 'Absent', 'Late', or NULL (not yet marked).
- **timestamp (DATETIME, DEFAULT CURRENT_TIMESTAMP):**
  - When the record was created/updated.
- **recorded_by (INT, FOREIGN KEY):**
  - Links to Teachers.teacher_id (who marked it).

Constraints

- UNIQUE (student_id, session_id):
    - One attendance record per student per session.
- FOREIGN KEY constraints:
    - Deletes records if student or session is removed (ON DELETE CASCADE).
    - Sets recorded_by to NULL if the teacher is deleted (ON DELETE SET NULL).

Role in System

- Teachers mark attendance for their students.
- NULL default simplifies marking; only conducted sessions ('Completed') matter for calculations.

---

7. Student_Subject Table

```
CREATE TABLE Student_Subject (
   student_id INT,
   subject_id INT,
   PRIMARY KEY (student_id, subject_id),
   FOREIGN KEY (student_id) REFERENCES Students(student_id) ON DELETE CASCADE,
   FOREIGN KEY (subject_id) REFERENCES Subjects(subject_id) ON DELETE CASCADE
);
```

Purpose

Manages the many-to-many relationship between students and subjects (enrollment).

Columns

- **student_id (INT, FOREIGN KEY):**
    - Links to Students.student_id.
- **subject_id (INT, FOREIGN KEY):**
    - Links to Subjects.subject_id.

Constraints

- PRIMARY KEY (student_id, subject_id):
    - Ensures no duplicate enrollments.
- FOREIGN KEY with ON DELETE CASCADE:
    - Removes enrollments if a student or subject is deleted.

Role in System

- Filters students visible to a teacher (via Subjects.teacher_id).
- Determines who gets pre-populated in Attendance.

---

8. Calendar_Exceptions Table (Optional)

```
CREATE TABLE Calendar_Exceptions (
   exception_id INT PRIMARY KEY AUTO_INCREMENT,
   date DATE NOT NULL,
   description VARCHAR(100),
   UNIQUE (date)
);
```

Purpose

Tracks holidays or other exceptions affecting session status.

Columns

- **exception_id (INT, PRIMARY KEY, AUTO_INCREMENT)**:
    - Unique ID for each exception.
- **date (DATE, NOT NULL)**:
    - Date of the exception (e.g., "2025-03-15").
- **description (VARCHAR(100))**:
    - Reason (e.g., "Spring Break").

Constraints

- UNIQUE (date):
    - Prevents duplicate exceptions on the same day.

Role in System

- Used to set Sessions.status to 'Cancelled' for holidays, excluding them from attendance totals.

How It All Ties Together

1. **Teacher Logs In**: Uses Teachers.email and pass.
2. **Sees Calendar**: Timetable (weekly schedule) and Sessions (specific dates) populate the view.
3. **Marks Attendance**:
    - Queries Student_Subject and Subjects to list students for a session.
    - Updates Attendance.status for their students in a 'Completed' session.
4. **Holidays**: Calendar_Exceptions marks sessions as 'Cancelled'.
5. **Attendance Calculation**: Counts 'Present' vs. total 'Completed' sessions.

Final Notes

This schema is **robust and complete** for your needs:

- **Scalable**: Handles multiple teachers, students, and subjects.
- **Flexible**: Supports calendar views and holiday adjustments.
- **Secure**: Enforces teacher-specific access via relationships.