DBMS Project

Parental Engagement Platform

that displays their child's schedule, grades, upcoming events, and school announcements

Made by:
Soham Marathe (A180)
Sakshi Bhure (A164)

Index

Introduction

The Parental Engagement Platform is designed to facilitate seamless communication and engagement between parents, teachers, and schools. This platform empowers parents and teachers with essential tools to support student learning and stay connected with their educational journey.

How It Works:

Parents and teachers begin by creating their accounts with unique user ids.

Dashboard Overview:
The dashboard serves as a central hub, offering convenient access to key features:

View Child's/Student's Information: Parents can easily access and view their child's/student's information, including academic performance, attendance, and extracurricular activities.

Schedule Parent-Teacher Meetings:
Parents: Schedule one-on-one meetings with teachers to discuss their child's progress, strengths, and areas for improvement.
Teachers: Set available time slots for parent-teacher meetings, allowing parents to schedule appointments conveniently.

Participate in Events:

Parents: RSVP for upcoming school events such as parent-teacher conferences, school performances, or fundraisers directly from the platform.
Teachers: Create and manage school events, allowing parents to view details and confirm attendance.

View Syllabus:

Parents: Access the syllabus for each of their child's classes, providing insight into the curriculum and topics covered.
Teachers: Share syllabi with parents, detailing course objectives, assignments, and important dates.
Key Features for Parents and Teachers:

Benefits:

Efficient Communication: Streamline communication between parents and teachers, fostering collaboration and shared goals for student success.

Empowered Engagement: Parents are empowered with timely information and tools to actively participate in their child's/student's education.

Transparency and Insight: Access to grades, schedules, and syllabi offers transparency, enabling informed decisions and supportive actions.

Entities and relations

The entities and their respective attributes are:

Parent
Parent_id (PK)
parent_Fname
parent_Lname
Parent_contact_num

Student
Student_id (PK)
student_Fname
student_Lname
student_class
student_extracurr
student_med_info
Student_aca_perf

Teacher
Teacher_id (PK)
teacher_Fname
teacher_Lname
teacher_position
teacher_prof_bio
teacher_email
Teacher_ph_no

School
School_id (PK)
school_name
school_email
school_contact_num

Subject
subject_id(PK)
subject_name
subject_credits

Syllabus
syllabus_id(PK)
unit_num
chapter_name
Weightage

Class
Class_id (PK)
class_division
class_teacher
class_room

Event
Event_id(PK)
event_name
event_organiser
event_description
event_date
event_timing
Event_venue

Schedule
schedule_id(PK)
from_till
subject_scheduled

Address
address_id (PK)
building_name
sector_num
city
state
pincode
landmark

Bus
bus_id(PK)
bus_driver_contact
Route_id (FK)

Route
route_id(PK)
route_name
Sector_num

The above entities are related to each other in the following manner:

- Parent-Has-Student
- Parent-Has-Address
- Student-Has-Parent
- Student-Has-School
- Student-Has-Class
- Student-Has-Address
- Teacher-Teaches-Subject
- Teacher-Works-At-School
- Teacher-Teaches-Class
- Teacher-Drives-Bus
- School-Has-Class
- School-Organises-Event
- School-Has-Route
- Subject-Has-Syllabus
- Syllabus-Belongs-To-Subject
- Route-Has-Bus
- Route-Has-Address
- Event-Held-In-School
- Event-Scheduled-In-School
- Schedule-For-Event
- Schedule-Includes-Subject

After all the relations have been derived, the entities will gain some foreign keys and even new tables may get created as follows:

parent
Parent_id (PK)
parent_Fname
parent_Lname
Parent_contact_num
Address_id (FK)

student
Student_id (PK)
student_Fname
student_Lname
student_class
student_extracurr
student_med_info
Student_aca_perf
Parent_id (FK)
School_id (FK)
Class_id (FK)
Address_id (FK)

teacher
Teacher_id (PK)
teacher_Fname
teacher_Lname
teacher_position
teacher_prof_bio
teacher_email
Teacher_ph_no
School_id (FK)
Class_id (FK)

Student_teacher
Student_id (PK)
Teacher_id
Subject_name

Teacher_subject
Teacher_id (PK)
Subject_id

school
School_id (PK)
school_name
school_email
school_contact_num

subject
subject_id(PK)
subject_name
subject_credits

syllabus
syllabus_id(PK)
unit_num
chapter_name
Weightage
subject_id(FK)

class
Class_id (PK)
class_division
class_teacher
class_room

event
Event_id(PK)
event_name
event_organiser
event_description
event_date
event_timing
Event_venue
School_id (FK)

schedule
schedule_id(PK)
from_till
subject_scheduled

address
address_id (PK)
building_name
sector_num
city
state
pincode
landmark

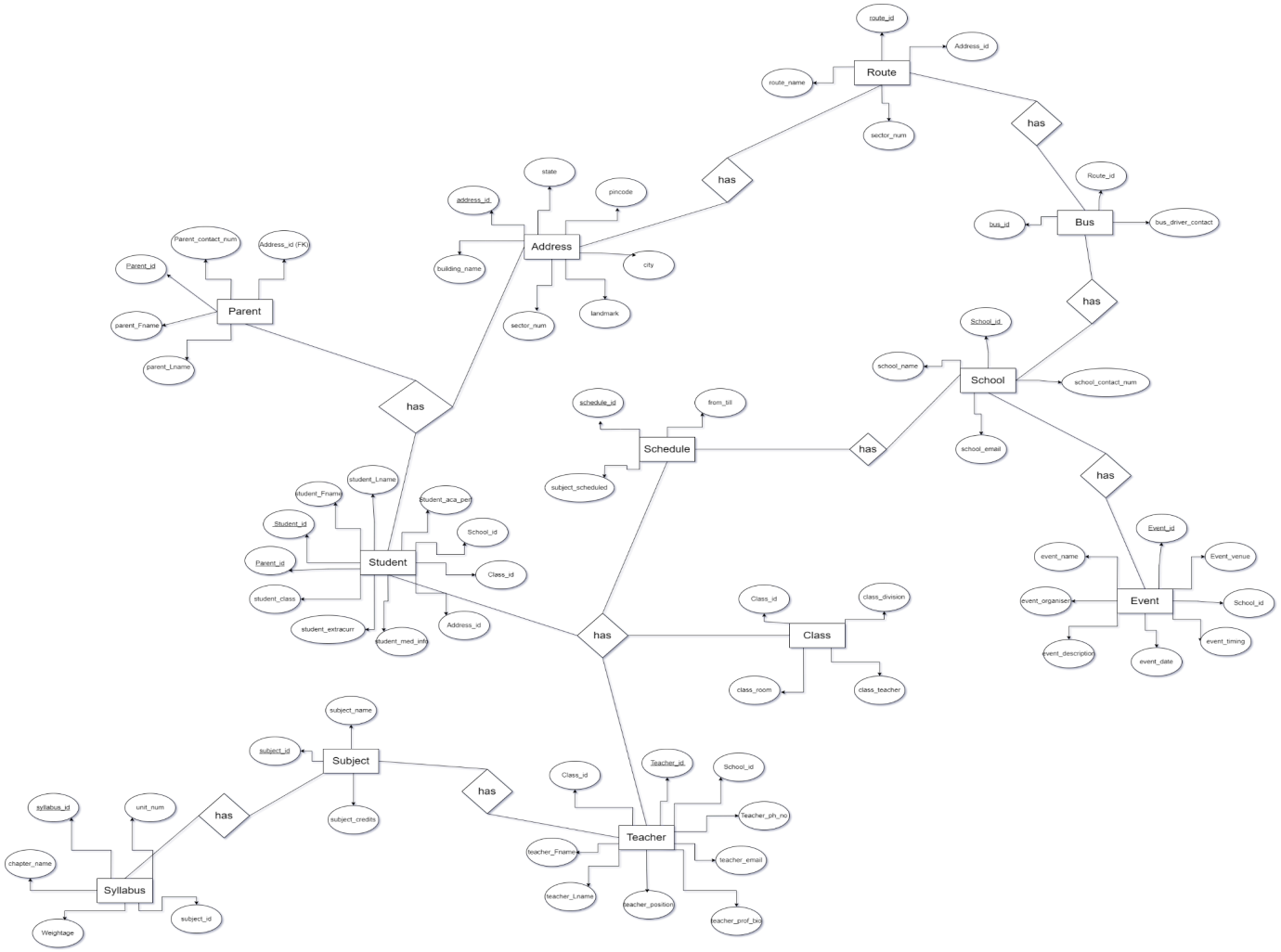
bus
bus_id(PK)
bus_driver_contact
Route_id (FK)

route
route_id(PK)
route_name
sector_num
Address_id (FK)

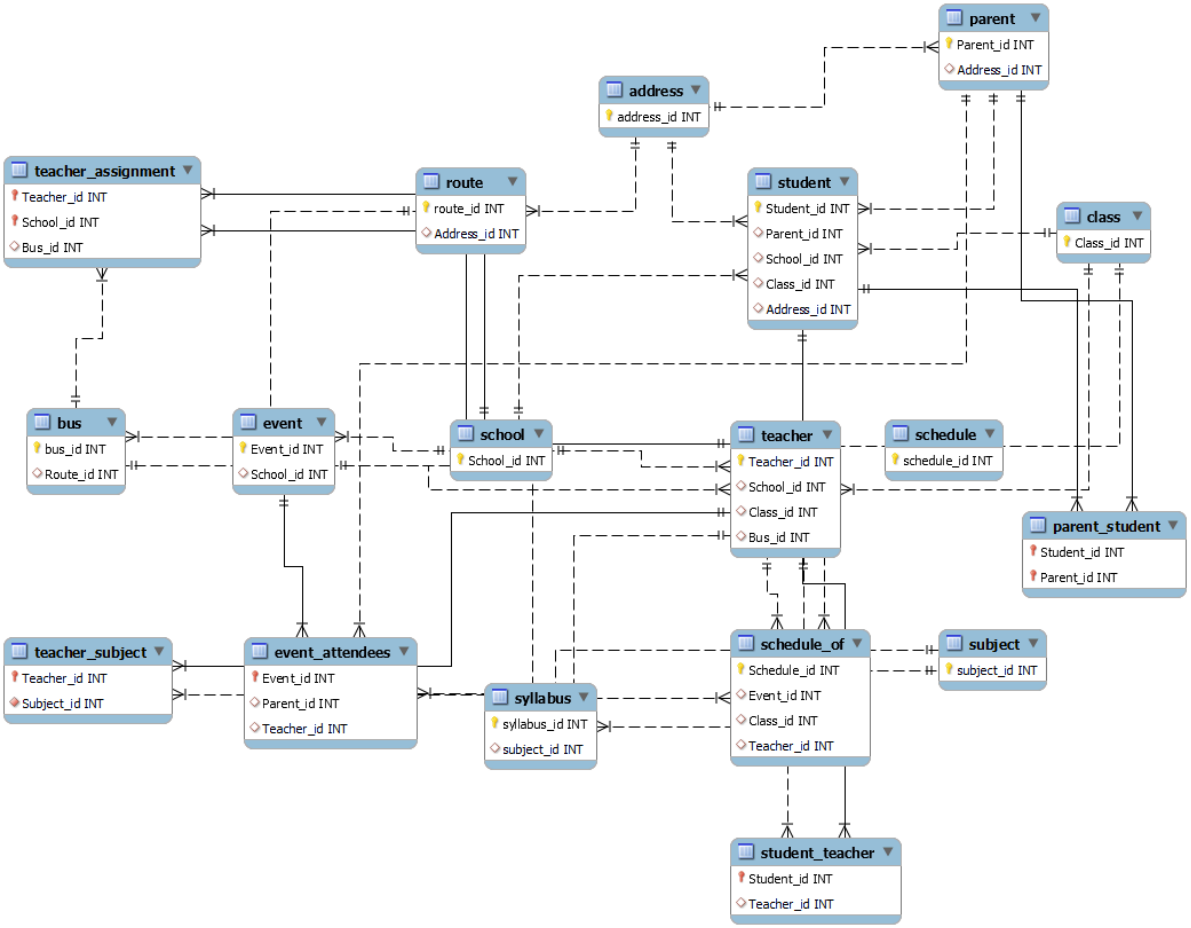
Event_attendees
Event_id (PK)
Parent_id
Teacher_id
Role

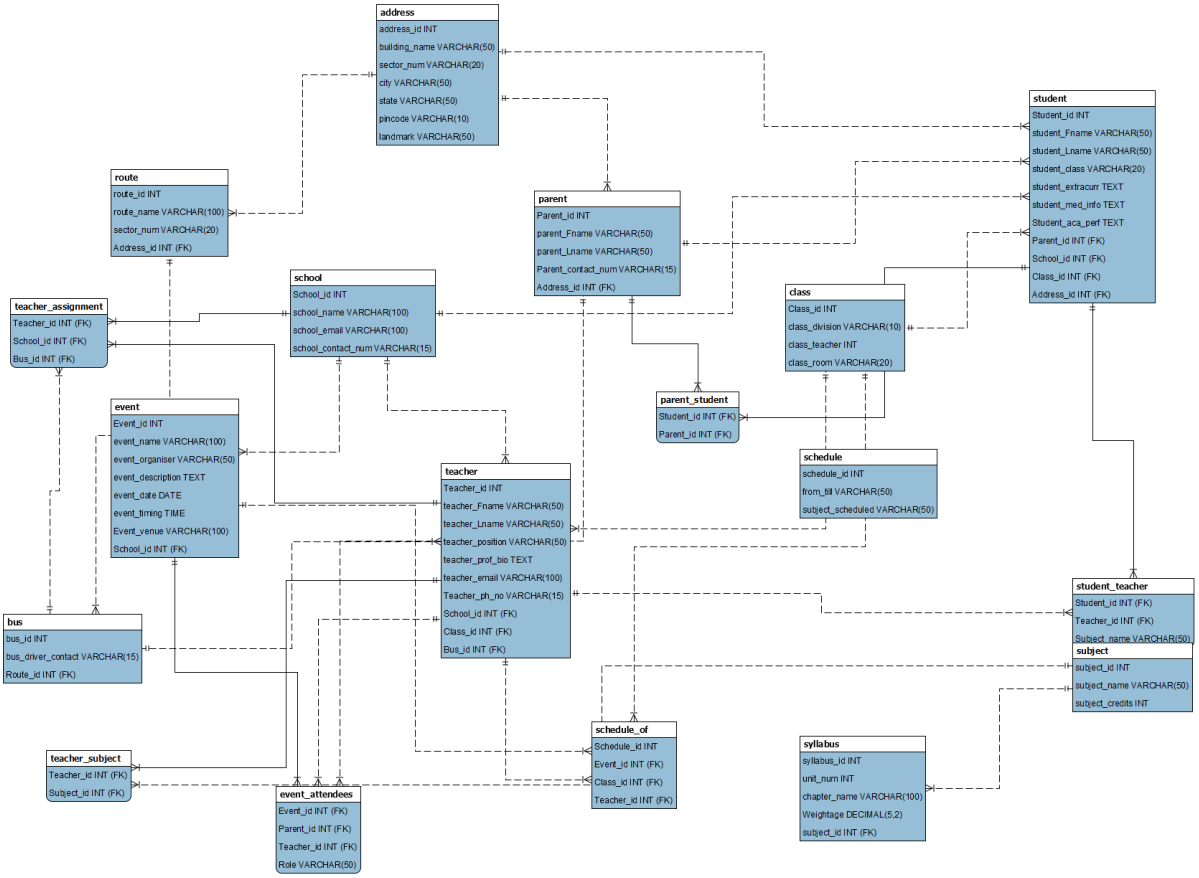
Schedule_of
Schedule_id (PK)
Event_id
Class_id
teacher_id

ER Diagram


route_id
Address_id
route_name
Route
sector_num
has
has
state
pincode
address_id
Address
building_name
city
sector_num
landmark
Route_id
bus_id
Bus
bus_driver_contact
has
School_id
school_name
School
school_contact_num
school_email
has
Parent_contact_num
Address_id (FK)
Parent_id
Parent
parent_Fname
parent_Lname
has
schedule_id
from_till
Schedule
subject_scheduled
has
has
Event_id
event_name
Event_venue
event_organiser
Event
School_id
event_description
event_date
event_timing
student_Fname
student_Lname
Student_id
student_aca_perf
School_id
Parent_id
Student
Class_id
student_class
student_extracurr
student_med_info
Address_id
has
Class_id
class_division
Class
class_room
class_teacher
subject_name
subject_id
Subject
subject_credits
has
has
Class_id
Teacher_id
School_id
Teacher_ph_no
Teacher
teacher_Fname
teacher_email
teacher_Lname
teacher_position
teacher_prof_bio
syllabus_id
unit_num
chapter_name
Syllabus
Weightage
subject_id

Relational model




address
address_id INT
building_name VARCHAR(50)
sector_num VARCHAR(20)
city VARCHAR(50)
state VARCHAR(50)
pincode VARCHAR(10)
landmark VARCHAR(50)
student
Student_id INT
student_Fname VARCHAR(50)
student_Lname VARCHAR(50)
student_class VARCHAR(20)
student_extracurr TEXT
student_med_info TEXT
Student_aca_perf TEXT
Parent_id INT (FK)
School_id INT (FK)
Class_id INT (FK)
Address_id INT (FK)
route
route_id INT
route_name VARCHAR(100)
sector_num VARCHAR(20)
Address_id INT (FK)
parent
Parent_id INT
parent_Fname VARCHAR(50)
parent_Lname VARCHAR(50)
Parent_contact_num VARCHAR(15)
Address_id INT (FK)
school
School_id INT
school_name VARCHAR(100)
school_email VARCHAR(100)
school_contact_num VARCHAR(15)
class
Class_id INT
class_division VARCHAR(10)
class_teacher INT
class_room VARCHAR(20)
teacher_assignment
Teacher_id INT (FK)
School_id INT (FK)
Bus_id INT (FK)
parent_student
Student_id INT (FK)
Parent_id INT (FK)
event
Event_id INT
event_name VARCHAR(100)
event_organiser VARCHAR(50)
event_description TEXT
event_date DATE
event_timing TIME
Event_venue VARCHAR(100)
School_id INT (FK)
schedule
schedule_id INT
from_till VARCHAR(50)
subject_scheduled VARCHAR(50)
teacher
Teacher_id INT
teacher_Fname VARCHAR(50)
teacher_Lname VARCHAR(50)
teacher_position VARCHAR(50)
teacher_prof_bio TEXT
teacher_email VARCHAR(100)
Teacher_ph_no VARCHAR(15)
School_id INT (FK)
Class_id INT (FK)
Bus_id INT (FK)
student_teacher
Student_id INT (FK)
Teacher_id INT (FK)
Subject_name VARCHAR(50)
bus
bus_id INT
bus_driver_contact VARCHAR(15)
Route_id INT (FK)
subject
subject_id INT
subject_name VARCHAR(50)
subject_credits INT
schedule_of
Schedule_id INT
Event_id INT (FK)
Class_id INT (FK)
Teacher_id INT (FK)
teacher_subject
Teacher_id INT (FK)
Subject_id INT (FK)
syllabus
syllabus_id INT
unit_num INT
chapter_name VARCHAR(100)
Weightage DECIMAL(5,2)
subject_id INT (FK)
event_attendees
Event_id INT (FK)
Parent_id INT (FK)
Teacher_id INT (FK)
Role VARCHAR(50)

Functional Dependencies

Given the above relations, we derive the following functional dependencies:

Parent_id -> Parent_Fname, Parent_Lname, Parent_contact_num

Address_id -> building_name, sector_num, city, state, pincode, landmark

Parent_id -> Address_id

Student_id -> student_Fname, student_Lname, student_class, student_extracurr, student_med_info, Student_aca_perf

Student_id -> parent_id, School_id, Class_id, Address_id

Teacher_id -> teacher_Fname, teacher_Lname, teacher_position, teacher_prof_bio, teacher_email, Teacher_ph_no

Teacher_id -> School_id, Class_id

School_id -> school_name, school_email, school_contact_num

Subject_id -> subject_name, subject_credits

Syllabus_id -> unit_num, chapter_name, Weightage

Subject_id -> Syllabus_id

Class_id -> class_division, class_room, Class_teacher

Class_id -> Schedule_id

Event_id -> event_name, event_organiser, event_description, event_date, event_timing, Event_venue

Event_id -> School_id

Schedule_id -> from_till, subject_scheduled

Bus_id -> Bus_driver_contact

Route_id -> Bus_id

Route_id -> route_name, sector_number

Address_id -> Route_id

Therefore, to find the candidate key:

Closure of Parent_id: {Parent_id, Parent_Fname, Parent_Lname, Parent_contact_num, Address_id}

Closure of Address_id: {Address_id, building_name, sector_num, city, state, pincode, landmark}

Closure of Student_id: {Student_id, student_Fname, student_Lname, student_class, student_extracurr, student_med_info, Student_aca_perf, parent_id, School_id, Class_id, Address_id}

Closure of Teacher_id: {Teacher_id, teacher_Fname, teacher_Lname, teacher_position, teacher_prof_bio, teacher_email, Teacher_ph_no, School_id, Class_id}

Closure of School_id: {School_id, school_name, school_email, school_contact_num}

Closure of Subject_id: {Subject_id, subject_name, subject_credits, Syllabus_id}

Closure of Syllabus_id: {Syllabus_id, unit_num, chapter_name, Weightage}

Closure of Class_id: {Class_id, class_division, class_room, Class_teacher, Schedule_id}

Closure of Event_id: {Event_id, event_name, event_organiser, event_description, event_date, event_timing, Event_venue,

Closure of Schedule_id: {Schedule_id, from_till, subject_scheduled}

Closure of Bus_id: {Bus_id, Bus_driver_contact}

Closure of Route_id: {Route_id, route_name, sector_number, Address_id}

Therefore, the candidate key is:

(Student_id, Teacher_id, Subject_id, Event_id)

The prime attributes are:

- Student_id
- Teacher_id
- Subject_id
- Event_id

Normalisation

First Normal Form (1NF):

A relation is in 1NF if and only if all attributes are atomic (indivisible) and there are no multi-valued or composite attributes allowed.

Since all attributes are single valued in our relational database, it is in 1NF.

Second Normal Form (2NF):

A relation is in 2NF if it is in 1NF and every non-prime attribute is fully functionally dependent on the entire candidate key.

There should be no partial dependency of non-prime attributes on the candidate key.

- For *Student (Student_id, student_Fname, student_Lname, student_class, student_extracurr, student_med_info, Student_aca_perf, parent_id, School_id, Class_id, Address_id)* :

  Candidate Key: Student_id

This table violates 2NF. The attribute Parent_id depends on the Student_Lname (assuming a child inherits the last name from one parent). This creates a partial dependency, where Parent_id is not dependent on the entire primary key (Student_id).

So, create a separate table called **Parent_Student** with primary key as Parent_id and foreign key as Student_id.

- For *Teacher (Teacher_id, teacher_Fname, teacher_Lname, teacher_position, teacher_prof_bio, teacher_email, TeacheCandidate Key: Teacher_idr_ph_no, School_id, Class_id)* :

  Candidate Key: Teacher_id

  Not in 2NF because attributes like teacher_Fname, teacher_Lname, etc., depend on Teacher_id but not on the entire candidate key (which includes School_id, Class_id).

  So, create a separate table called **teacher_assignment** having Teacher_id, School_id, Class_id.

  Third Normal Form (3NF):

  A relation is in 3NF if it is in 2NF and no transitive dependencies exist.

  Transitive dependency occurs when a non-prime attribute depends on another non-prime attribute, which itself depends on the candidate key.

  As the dependencies have no transitive dependencies, they are already in 3NF.

Boyce-Codd Normal Form (BCNF):

A relation is in BCNF if for every one of its dependencies X → Y, X is a superkey.

Essentially, it's a stricter form of 3NF where every functional dependency determines a candidate key.

All the relations are already in BCNF. Every non-trivial functional dependency in these relations has a superkey on the left-hand side. So, no further decomposition is needed for BCNF compliance.

Queries

```
CREATE TABLE parent (
    Parent_id INT not null,
    primary key(Parent_id),
    parent_Fname VARCHAR(50),
    parent_Lname VARCHAR(50),
    Parent_contact_num VARCHAR(15),
    Address_id INT,
    FOREIGN KEY (Address_id) REFERENCES address(address_id));

INSERT INTO parent (Parent_id, parent_Fname, parent_Lname,
Parent_contact_num, Address_id)
VALUES
(1, 'John', 'Doe', '1234567890', 1),
(2, 'Jane', 'Smith', '9876543210', 2),
(3, 'Michael', 'Johnson', '5555555555', 3),
(4, 'Emily', 'Williams', '3333333333', 4),
(5, 'David', 'Brown', '4444444444', 5);

CREATE TABLE student (
    Student_id INT not null,
    primary key(student_id),
    student_Fname VARCHAR(50),
    student_Lname VARCHAR(50),
    student_class VARCHAR(20),
```

```
    student_extracurr TEXT,
    student_med_info TEXT,
    Student_aca_perf TEXT,
    Parent_id INT,
    School_id INT,
    Class_id INT,
    Address_id INT,
    FOREIGN KEY (Parent_id) REFERENCES parent(Parent_id),
    FOREIGN KEY (School_id) REFERENCES school(School_id),
    FOREIGN KEY (Class_id) REFERENCES class(Class_id),
    FOREIGN KEY (Address_id) REFERENCES address(address_id)
);

INSERT INTO student (Student_id, student_Fname, student_Lname, student_class,
student_extracurr, student_med_info, Student_aca_perf, Parent_id, School_id,
Class_id, Address_id)
VALUES
(1, 'Alice', 'Doe', 'Grade 9', 'Sports Club', 'None', 'A+', 1, 1, 1, 1),
(2, 'Bob', 'Smith', 'Grade 10', 'Chess Club', 'Asthma', 'B+', 2, 1, 2, 2),
(3, 'Ella', 'Johnson', 'Grade 8', 'Art Club', 'Allergic to nuts', 'A', 3, 2, 3, 3),
(4, 'Tom', 'Williams', 'Grade 11', 'Music Band', 'None', 'A-', 4, 2, 4, 4),
(5, 'Sophia', 'Brown', 'Grade 7', 'Drama Club', 'None', 'B', 5, 3, 5, 5);

CREATE TABLE teacher (
    Teacher_id INT not null,
 Primary key (Teacher_id ),
```

```
    teacher_Fname VARCHAR(50),
    teacher_Lname VARCHAR(50),
    teacher_position VARCHAR(50),
    teacher_prof_bio TEXT,
    teacher_email VARCHAR(100),
    Teacher_ph_no VARCHAR(15),
    School_id INT,
    Class_id INT,
    Bus_id INT,
    FOREIGN KEY (School_id) REFERENCES school(School_id),
    FOREIGN KEY (Class_id) REFERENCES class(Class_id),
    FOREIGN KEY (Bus_id) REFERENCES bus(bus_id)
);

INSERT INTO teacher (Teacher_id, teacher_Fname, teacher_Lname,
teacher_position, teacher_prof_bio, teacher_email, Teacher_ph_no, School_id,
Class_id, Bus_id)
VALUES
(1, 'Mr.', 'Anderson', 'Math Teacher', 'Experienced math teacher with a passion for
learning.', 'mr.anderson@school.com', '123-456-7890', 1, 1, 1),
(2, 'Ms.', 'Taylor', 'English Teacher', 'Lover of literature and writing.',
'ms.taylor@school.com', '987-654-3210', 1, 2, 2),
(3, 'Mr.', 'Clark', 'Science Teacher', 'Enthusiastic about exploring the wonders of the
universe.', 'mr.clark@school.com', '555-555-5555', 2, 3, 3),
(4, 'Ms.', 'Roberts', 'History Teacher', 'Bringing history to life with engaging lessons.',
'ms.roberts@school.com', '333-333-3333', 2, 4, 4),
```

```
(5, 'Mr.', 'Garcia', 'Art Teacher', 'Inspiring creativity and imagination.',
'mr.garcia@school.com', '444-444-4444', 3, 5, NULL);

CREATE TABLE student_teacher (
    Student_id INT not null,
    Teacher_id INT,
    Subject_name VARCHAR(50),
    PRIMARY KEY (Student_id),
    FOREIGN KEY (Student_id) REFERENCES student(Student_id),
    FOREIGN KEY (Teacher_id) REFERENCES teacher(Teacher_id)
);

INSERT INTO student_teacher (Student_id, Teacher_id, Subject_name)
VALUES
(1, 1, 'Math'),
(2, 2, 'English'),
(3, 3, 'Science'),
(4, 4, 'History'),
(5, 5, 'Art');

CREATE TABLE teacher_subject (
    Teacher_id INT,
    Subject_id INT not null,
    PRIMARY KEY (Teacher_id),
    FOREIGN KEY (Teacher_id) REFERENCES teacher(Teacher_id),
    FOREIGN KEY (Subject_id) REFERENCES subject(subject_id)
```

```
);

INSERT INTO teacher_subject (Teacher_id, Subject_id)
VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

CREATE TABLE school (
    School_id INT not null,
PRIMARY KEY(School_id),
    school_name VARCHAR(100),
    school_email VARCHAR(100),
    school_contact_num VARCHAR(15)
);

INSERT INTO school (School_id, school_name, school_email, school_contact_num)
VALUES
(1, 'Oakridge High School', 'info@oakridge.com', '111-111-1111'),
(2, 'Maple Leaf Academy', 'info@mapleleaf.com', '222-222-2222'),
(3, 'Pine Crest Elementary', 'info@pinecrest.com', '333-333-3333');

CREATE TABLE subject (
```

```
    subject_id INT not null,
   Primary key(subject_id),
    subject_name VARCHAR(50),
    subject_credits INT
);

INSERT INTO subject (Subject_id, subject_name, subject_credits)
VALUES
(1, 'Math', 4),
(2, 'English', 3),
(3, 'Science', 3),
(4, 'History', 3),
(5, 'Art', 2);


CREATE TABLE syllabus (
    syllabus_id INT not null,
    Primary key(syllabus_id),
    unit_num INT,
    chapter_name VARCHAR(100),
    Weightage DECIMAL(5,2),
    subject_id INT,
    FOREIGN KEY (subject_id) REFERENCES subject(subject_id)
);
```

```
INSERT INTO syllabus (syllabus_id, unit_num, chapter_name, Weightage,
subject_id)
VALUES
(1, 1, 'Algebra', 0.2, 1),
(2, 1, 'Literature', 0.3, 2),
(3, 1, 'Chemical Reactions', 0.25, 3),
(4, 1, 'Ancient Civilizations', 0.3, 4),
(5, 1, 'Drawing Basics', 0.2, 5);

CREATE TABLE class (
    Class_id INT not null,
 PRIMARY KEY(class_id),
    class_division VARCHAR(10),
    class_teacher INT,
    class_room VARCHAR(20)
);

INSERT INTO class (Class_id, class_division, class_teacher, class_room)
VALUES
(1, '9A', 1, 'Room 101'),
(2, '10B', 2, 'Room 202'),
(3, '8C', 3, 'Room 303'),
(4, '11D', 4, 'Room 404'),
(5, '7E', 5, 'Art Room');

CREATE TABLE event (
```

```
    Event_id INT not null,
PRIMARY KEY(event_id),
    event_name VARCHAR(100),
    event_organiser VARCHAR(50),
    event_description TEXT,
    event_date DATE,
    event_timing TIME,
    Event_venue VARCHAR(100),
    School_id INT,
    FOREIGN KEY (School_id) REFERENCES school(School_id)
);

INSERT INTO event (Event_id, event_name, event_organiser, event_description,
event_date, event_timing, Event_venue, School_id)
VALUES
(1, 'Science Fair', 'School', 'Annual science fair showcasing student projects.',
'2024-05-15', '10:00', 'School Hall', 1),
(2, 'Literary Evening', 'School', 'Celebration of literature with readings and
performances.', '2024-06-20', '6:00', 'Library', 2),
(3, 'Art Exhibition', 'School', 'Display of student artwork from the semester.',
'2024-07-10', '2:00', 'Art Room', 3),
(4, 'Parent-Teacher Meeting', 'Teacher', 'Scheduled parent-teacher meetings for
progress updates.', '2024-04-25', '3:00', 'School Hall', 1),
(5, 'Math Quiz Competition', 'Teacher', 'Inter-class math quiz competition.',
'2024-04-30', '11:00', 'Room 101', 1);
```

```
select * from event;
CREATE TABLE schedule (
    schedule_id INT not null,
PRIMARY KEY(schedule_id),
    from_till VARCHAR(50),
    subject_scheduled VARCHAR(50)
);

INSERT INTO schedule (schedule_id, from_till, subject_scheduled)
VALUES
(1, '9:00 AM - 10:30 AM', 'Math'),
(2, '10:45 AM - 12:15 PM', 'English'),
(3, '1:00 PM - 2:30 PM', 'Science'),
(4, '9:00 AM - 10:30 AM', 'History'),
(5, '10:45 AM - 12:15 PM', 'Art');


CREATE TABLE address (
    address_id INT not null,
PRIMARY KEY(address_id),
    building_name VARCHAR(50),
    sector_num VARCHAR(20),
    city VARCHAR(50),
    state VARCHAR(50),
    pincode VARCHAR(10),
    landmark VARCHAR(50)
```

```
);

INSERT INTO address (address_id, building_name, sector_num, city, state, pincode,
landmark)
VALUES
(1, '123 Main St', 'Sector A', 'Cityville', 'State A', '12345', 'Near Park'),
(2, '456 Oak Ave', 'Sector B', 'Townville', 'State B', '54321', 'Opposite Library'),
(3, '789 Maple Rd', 'Sector C', 'Villagetown', 'State C', '67890', 'Next to School'),
(4, '101 Pine Lane', 'Sector D', 'Forest City', 'State D', '98765', 'By River'),
(5, '111 Elm Street', 'Sector E', 'Gardenvale', 'State E', '45678', 'Corner Store');



CREATE TABLE bus (
   bus_id INT not null,
PRIMARY KEY(bus_id),
   bus_driver_contact VARCHAR(15),
   Route_id INT,
   FOREIGN KEY (Route_id) REFERENCES route(route_id)
);

INSERT INTO bus (bus_id, bus_driver_contact, Route_id)
VALUES
(1, '111-222-3333', 1),
(2, '444-555-6666', 2),
(3, '777-888-9999', 3),
(4, '123-456-7890', 4),
```

```
(5, '987-654-3210', 5);



CREATE TABLE route (
    route_id INT not null,
 PRIMARY KEY(route_id),
    route_name VARCHAR(100),
    sector_num VARCHAR(20),
    Address_id INT,
    FOREIGN KEY (Address_id) REFERENCES address(address_id)
);

INSERT INTO route (route_id, route_name, sector_num, Address_id)
VALUES
(1, 'School Route 1', 'Sector A - B', 1),
(2, 'School Route 2', 'Sector C - D', 2),
(3, 'School Route 3', 'Sector E - F', 3),
(4, 'School Route 4', 'Sector G - H', 4),
(5, 'School Route 5', 'Sector I - J', 5);

CREATE TABLE Event_attendees (
    Event_id INT not null,
Primary key(Event_id),
    Parent_id INT,
    Teacher_id INT,
    Role VARCHAR(50),
```

```
    FOREIGN KEY (Event_id) REFERENCES event(Event_id),
    FOREIGN KEY (Parent_id) REFERENCES parent(Parent_id),
    FOREIGN KEY (Teacher_id) REFERENCES teacher(Teacher_id)
);

INSERT INTO Event_attendees (Event_id, Parent_id, Teacher_id, Role)
VALUES
(1, 1, NULL, 'Parent'),
(2, NULL, 1, 'Teacher'),
(3, NULL, 5, 'Teacher'),
(4, 3, 3, 'Both'),
(5, NULL, 1, 'Teacher');

CREATE TABLE Schedule_of (
    Schedule_id INT not null,
Primary key(Schedule_id),
    Event_id INT,
    Class_id INT,
    Teacher_id INT,
    FOREIGN KEY (Event_id) REFERENCES event(Event_id),
    FOREIGN KEY (Class_id) REFERENCES class(Class_id),
    FOREIGN KEY (Teacher_id) REFERENCES teacher(Teacher_id)
);

INSERT INTO Schedule_of (Schedule_id, Event_id, Class_id, Teacher_id)
VALUES
```

```
(1, 1, 1, 1),
(2, 2, 2, 2),
(3, 3, 3, 3),
(4, 4, 4, 4),
(5, 5, 5, 1);

CREATE TABLE Parent_Student (
    Student_id INT not null,
    Parent_id INT not null,
    FOREIGN KEY (Student_id) REFERENCES Student(Student_id),
    FOREIGN KEY (Parent_id) REFERENCES Parent(Parent_id),
    PRIMARY KEY (Student_id, Parent_id)  /* Composite Primary Key */
);

INSERT INTO Parent_Student (Parent_id, Student_id)
VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

CREATE TABLE Teacher_Assignment (
    Teacher_id INT not null,
    School_id INT not null,
    Bus_id INT,
```

```
    FOREIGN KEY (Teacher_id) REFERENCES Teacher(Teacher_id),
    FOREIGN KEY (School_id) REFERENCES school(School_id),
    FOREIGN KEY (Bus_id) REFERENCES bus(bus_id),
    PRIMARY KEY (Teacher_id, School_id)  /* Composite Primary Key */
);

INSERT INTO Teacher_Assignment (Teacher_id, School_id, Bus_id)
VALUES
(1, 1, 1),
(2, 1, 2),
(3, 2, 3),
(4, 2, 4),
(5, 3, NULL);

-- 1. Parent Can View Student's Information:
-- Assuming Parent ID 1 wants to view their student's information
SELECT s.Student_id, s.student_Fname, s.student_Lname, s.student_class,
s.student_extracurr, s.student_med_info, s.Student_aca_perf
FROM Student s
INNER JOIN Parent_Student ps ON s.Student_id = ps.Student_id
INNER JOIN Parent p ON ps.Parent_id = p.Parent_id
WHERE p.Parent_id = 1;

-- 2. Teacher Can View Student's Information:
-- Assuming Teacher ID 1 wants to view their student's information
```

```
SELECT s.Student_id, s.student_Fname, s.student_Lname, s.student_class,
s.student_extracurr, s.student_med_info, s.Student_aca_perf
FROM Student s
INNER JOIN student_teacher st ON s.Student_id = st.Student_id
INNER JOIN Teacher t ON st.Teacher_id = t.Teacher_id
WHERE t.Teacher_id = 1;

-- 3. Student Can View Its Schedule:
-- Assuming Student ID 1 wants to view their schedule
SELECT sch.from_till, sch.subject_scheduled
FROM Schedule sch
INNER JOIN Schedule_of so ON sch.schedule_id = so.Schedule_id
INNER JOIN Student s ON so.Student_id = s.Student_id
WHERE s.Student_id = 1;

-- 4. Teacher Can View Their Schedule:
-- Assuming Teacher ID 1 wants to view their schedule
SELECT sch.from_till, sch.subject_scheduled
FROM Schedule sch
INNER JOIN Schedule_of so ON sch.schedule_id = so.Schedule_id
INNER JOIN Teacher t ON so.Teacher_id = t.Teacher_id
WHERE t.Teacher_id = 1;

-- 5. Parent Can Schedule a Parent-Teacher Meeting:
-- Assuming the 'Event' table is used for Parent-Teacher meetings
```

```
INSERT INTO Event (event_name, event_organiser, event_description, event_date, event_timing, Event_venue, School_id)
VALUES ('Parent-Teacher Meeting', 'Parent', 'Scheduled parent-teacher meeting', '2024-04-25', '3:00 PM', 'School Hall', 1);
-- Assuming Parent attends the meeting
INSERT INTO Event_attendees (Event_id, Parent_id, Role)
VALUES (LAST_INSERT_ID(), 1, 'Parent');

-- 6. Teacher Can Schedule a Parent-Teacher Meeting:
-- Assuming the 'Event' table is used for Parent-Teacher meetings
INSERT INTO Event (event_name, event_organiser, event_description, event_date, event_timing, Event_venue, School_id)
VALUES ('Parent-Teacher Meeting', 'Teacher', 'Scheduled parent-teacher meeting', '2024-04-25', '3:00 PM', 'School Hall', 1);
-- Assuming Teacher attends the meeting
INSERT INTO Event_attendees (Event_id, Teacher_id, Role)
VALUES (LAST_INSERT_ID(), 1, 'Teacher');

-- 7. Parent, Teacher, and Student Can Participate in Events:
-- Assuming Parent participates in an event
INSERT INTO Event_attendees (Event_id, Parent_id, Role)
VALUES (1, 1, 'Parent');
-- Assuming Teacher participates in an event
INSERT INTO Event_attendees (Event_id, Teacher_id, Role)
VALUES (2, 1, 'Teacher');
-- Assuming Student participates in an event
```

INSERT INTO Event_attendees (Event_id, Student_id, Role)

VALUES (3, 1, 'Student');

```
332     -- 1. Parent Can View Student's Information:
333     -- Assuming Parent ID 1 wants to view their student's information
334 •   SELECT s.Student_id, s.student_Fname, s.student_Lname, s.student_class, s.student_extracurr, s.student_med_info, s.Student_aca_perf
335     FROM Student s
336     INNER JOIN Parent_Student ps ON s.Student_id = ps.Student_id
337     INNER JOIN Parent p ON ps.Parent_id = p.Parent_id
338     WHERE p.Parent_id = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Student_id	student_Fname	student_Lname	student_class	student_extracurr	student_med_info	Student_aca_perf
1	Alice	Doe	Grade 9	Sports Club	None	A+

```
340     -- 2. Teacher Can View Student's Information:
341     -- Assuming Teacher ID 1 wants to view their student's information
342 •   SELECT s.Student_id, s.student_Fname, s.student_Lname, s.student_class, s.student_extracurr, s.student_med_info, s.Student_aca_perf
343     FROM Student s
344     INNER JOIN student_teacher st ON s.Student_id = st.Student_id
345     INNER JOIN Teacher t ON st.Teacher_id = t.Teacher_id
346     WHERE t.Teacher_id = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Student_id	student_Fname	student_Lname	student_class	student_extracurr	student_med_info	Student_aca_perf
1	Alice	Doe	Grade 9	Sports Club	None	A+

Similarly ,we have included 5 other functionalities as follows:

```
-- 3. Student Can View Its Schedule:
-- Assuming Student ID 1 wants to view their schedule
SELECT sch.from_till, sch.subject_scheduled
FROM Schedule sch
INNER JOIN Schedule_of so ON sch.schedule_id = so.Schedule_id
INNER JOIN Student s ON so.Student_id = s.Student_id
WHERE s.Student_id = 1;

-- 4. Teacher Can View Their Schedule:
-- Assuming Teacher ID 1 wants to view their schedule
SELECT sch.from_till, sch.subject_scheduled
FROM Schedule sch
INNER JOIN Schedule_of so ON sch.schedule_id = so.Schedule_id
INNER JOIN Teacher t ON so.Teacher_id = t.Teacher_id
WHERE t.Teacher_id = 1;

-- 5. Parent Can Schedule a Parent-Teacher Meeting:
-- Assuming the 'Event' table is used for Parent-Teacher meetings
INSERT INTO Event (event_name, event_organiser, event_description, event_date,
event_timing, Event_venue, School_id)
VALUES ('Parent-Teacher Meeting', 'Parent', 'Scheduled parent-teacher meeting',
'2024-04-25', '3:00 PM', 'School Hall', 1);
-- Assuming Parent attends the meeting
INSERT INTO Event_attendees (Event_id, Parent_id, Role)
VALUES (LAST_INSERT_ID(), 1, 'Parent');
```

```
-- 6. Teacher Can Schedule a Parent-Teacher Meeting:
-- Assuming the 'Event' table is used for Parent-Teacher meetings
INSERT INTO Event (event_name, event_organiser, event_description, event_date, event_timing, Event_venue, School_id)
VALUES ('Parent-Teacher Meeting', 'Teacher', 'Scheduled parent-teacher meeting', '2024-04-25', '3:00 PM', 'School Hall', 1);
-- Assuming Teacher attends the meeting
INSERT INTO Event_attendees (Event_id, Teacher_id, Role)
VALUES (LAST_INSERT_ID(), 1, 'Teacher');

-- 7. Parent, Teacher, and Student Can Participate in Events:
-- Assuming Parent participates in an event
INSERT INTO Event_attendees (Event_id, Parent_id, Role)
VALUES (1, 1, 'Parent');
-- Assuming Teacher participates in an event
INSERT INTO Event_attendees (Event_id, Teacher_id, Role)
VALUES (2, 1, 'Teacher');
-- Assuming Student participates in an event
INSERT INTO Event_attendees (Event_id, Student_id, Role)
VALUES (3, 1, 'Student');
```