```
-- ================================================================
--      ACADEMIC INFORMATION SYSTEM SCHEMA (REFINED VERSION)
-- ================================================================
-- This script provides a complete and refined database schema for
-- a basic academic information system, optimized for Supabase.
--
-- Improvements include:
-- 1. English naming conventions for tables and columns.
-- 2. Optimized data types (TEXT, VARCHAR, and custom ENUM).
-- 3. Corrected syntax for all INSERT statements.
-- 4. Secure Row Level Security (RLS) policies.
-- ================================================================

-- ----------------------------------------------------
-- Section 1: Custom Type Definitions
-- ----------------------------------------------------
-- We create a custom type for grades to ensure data integrity.
-- This ensures that only valid grades can be inserted.

CREATE TYPE public.grade_options AS ENUM ('A', 'B', 'C', 'D', 'E');


-- ----------------------------------------------------
-- Section 2: Table Creation
-- ----------------------------------------------------

-- Table for lecturers/professors
CREATE TABLE public.lecturers (
    nip TEXT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    department VARCHAR(50),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Table for classes
CREATE TABLE public.classes (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    name VARCHAR(20) NOT NULL,
    year INT,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Table for students
-- Using TEXT for nim for flexibility.
CREATE TABLE public.students (
    nim TEXT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    major VARCHAR(50),
```

```
    year INT CHECK (year >= 2000),
    address TEXT,
    class_id UUID REFERENCES public.classes(id) ON DELETE SET NULL,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Table for courses
CREATE TABLE public.courses (
    code TEXT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    credits INT CHECK (credits > 0),
    lecturer_nip TEXT REFERENCES public.lecturers(nip) ON DELETE SET NULL,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Table for course schedules
CREATE TABLE public.schedules (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    course_code TEXT REFERENCES public.courses(code) ON DELETE CASCADE,
    day VARCHAR(10) CHECK (day IN ('Senin', 'Selasa', 'Rabu', 'Kamis', 'Jumat', 'Sabtu')),
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    room VARCHAR(20),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Table for student grades
-- This table links students and courses.
CREATE TABLE public.grades (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    student_nim TEXT REFERENCES public.students(nim) ON DELETE CASCADE,
    course_code TEXT REFERENCES public.courses(code) ON DELETE CASCADE,
    grade grade_options, -- Using the custom ENUM type here
    created_at TIMESTAMPTZ DEFAULT NOW()
);



-- ------------------------------------------------------
-- Section 3: Row Level Security (RLS)
-- ------------------------------------------------------
-- Activate RLS for all tables to enforce data access policies.
-- This is a crucial security step in Supabase.

ALTER TABLE public.lecturers ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.classes ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.students ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.courses ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.schedules ENABLE ROW LEVEL SECURITY;
```

```
ALTER TABLE public.grades ENABLE ROW LEVEL SECURITY;

-- Create policies: By default, no one can access the data.
-- The policy below allows any authenticated user to READ all data.
-- For production, you would create more granular policies.
-- For example, a student can only see their own grades.

CREATE POLICY "Allow authenticated read access"
ON public.lecturers FOR SELECT
USING (auth.role() = 'authenticated');

CREATE POLICY "Allow authenticated read access"
ON public.classes FOR SELECT
USING (auth.role() = 'authenticated');

CREATE POLICY "Allow authenticated read access"
ON public.students FOR SELECT
USING (auth.role() = 'authenticated');

CREATE POLICY "Allow authenticated read access"
ON public.courses FOR SELECT
USING (auth.role() = 'authenticated');

CREATE POLICY "Allow authenticated read access"
ON public.schedules FOR SELECT
USING (auth.role() = 'authenticated');

CREATE POLICY "Allow authenticated read access"
ON public.grades FOR SELECT
USING (auth.role() = 'authenticated');

-- Note: You should also add policies for INSERT, UPDATE, DELETE as needed.
-- Example:
-- CREATE POLICY "Students can insert their own grades"
-- ON public.grades FOR INSERT
-- WITH CHECK (auth.uid() = (SELECT user_id FROM profiles WHERE profiles.nim =
student_nim));



-- ------------------------------------------------------
-- Section 4: Seed Data (Sample Data Insertion)
-- ------------------------------------------------------
-- Inserting sample data with corrected syntax.

-- Insert lecturers
INSERT INTO public.lecturers (nip, name, department)
VALUES ('1978121200010001', 'Dr. Andi Prasetyo', 'Informatika');
```

```
-- Insert classes
INSERT INTO public.classes (name, year)
VALUES ('IF-1A', 2023);

-- Insert students (Corrected INSERT statements)
INSERT INTO public.students (nim, name, major, year, address, class_id)
VALUES
('2023001', 'Budi Santoso', 'Informatika', 2023, 'Jakarta', (SELECT id FROM public.classes
WHERE name = 'IF-1A' LIMIT 1)),
('2023002', 'Siti Aminah', 'Sistem Informasi', 2023, 'Bandung', NULL);

-- Insert courses (Corrected INSERT statements)
INSERT INTO public.courses (code, name, credits, lecturer_nip)
VALUES
('IF101', 'Algoritma dan Pemrograman', 3, '1978121200010001'),
('IF102', 'Basis Data', 3, '1978121200010001'),
('IF201', 'Pemrograman Web', 3, '1978121200010001');

-- Insert schedules
INSERT INTO public.schedules (course_code, day, start_time, end_time, room)
VALUES ('IF201', 'Senin', '08:00', '10:00', 'Ruang 101');

-- Insert grades (Corrected INSERT statements)
INSERT INTO public.grades (student_nim, course_code, grade)
VALUES
('2023001', 'IF101', 'A'),
('2023002', 'IF102', 'B');


-- ============================================================
--                    END OF SCRIPT
-- ============================================================
```