1) Introduction:-

The University Management System is designed to manage various aspects of a university's operations, encompassing faculties, students, professors, courses, enrollments, grades, rooms, examinations, fee payments, and scholarships. The system maintains detailed records for each entity, ensuring efficient and streamlined management of academic and administrative tasks. Faculties oversee different departments, with professors assigned to teach specific courses. Students enroll in these courses and their performance is tracked through grades. The system also schedules examinations, manages room allocations, processes fee payments, and awards scholarships based on academic performance. This comprehensive system aims to enhance the overall management and operational efficiency of the university.

2) System Requirements:-

- Miniworld: University environment
- Entities & Attributes:-
  - Faculties (faculty_id, faculty_name, department, email, phone_number)
    - Simple attributes: faculty_id, faculty_name, department, email, phone_number
  - Students (student_id, name {first_name, last_name}, email, date_of_birth, gender, address, major, gpa)
    - FK (faculty_id) refs Faculties
    - Simple Attributes: student_id, email, date_of_birth, gender, address, major
    - Composite Attributes: name
    - Derived Attributes: gpa
  - Professors (professor_id, name {first_name, last_name}, email, date_of_birth, gender, address, salary)
    - FK (faculty_id) refs Faculties
    - Simple Attributes: professor_id, email, date_of_birth, gender, address, salary
    - Composite Attributes: name
  - Courses (course_id, course_name, credits)
    - FK (faculty_id) refs Faculties
    - FK (professor_id) refs Professors
    - Simple Attributes: course_id, course_name, credits

- Enrollments (enrollment_id)
  - FK (student_id) refs Students
  - FK (course_id) refs Courses
  - Simple Attributes: enrollment_id

- Grades (grade_id, grade)
  - FK (enrollment_id) refs Enrollments
  - Simple Attributes: grade_id, grade
- Rooms (room_number, capacity, building, floor)
  - Simple Attributes: room_number, capacity, building, floor
- Examinations (exam_id, exam_date, start_time, end_time)
  - FK (location) refs Rooms
  - FK (course_id) refs Courses
  - Simple Attributes: exam_id, exam_date, start_time, end_time
- fee_payments (payment_id, payment_date, amount)
  - FK (student_id) refs Students
  - Simple Attributes: payment_id, payment_date, amount
- Scholarships (full_name, gpa)
  - FK (student_id) refs Students
  - Derived Attributes: full_name, gpa

3) System Design:-

Here we'll go through 3 different diagrams, we'll first start with Schema (Mapping) then we'll show the E-R Diagram and then lastly the UML class diagram.

- **Schema (Mapping):-**

Faculties

faculty_id	faculty_name	department	email	phone_number

Students

student_id	first_name	last_name	email	date_of_birth	gender	address	major	gpa	faculty_id

Professors

professor_id	first_name	last_name	email	date_of_birth	gender	address	salary	faculty_id

Courses

course_id	course_name	credits	faculty_id	professor_id

Enrollments

enrollment_id	student_id	course_id	enrollment_date

Grades

grade_id	enrollment_id	grade

Rooms

room_number	capacity	building	floor

Examinations

exam_id	course_id	exam_date	start_time	end_time	location

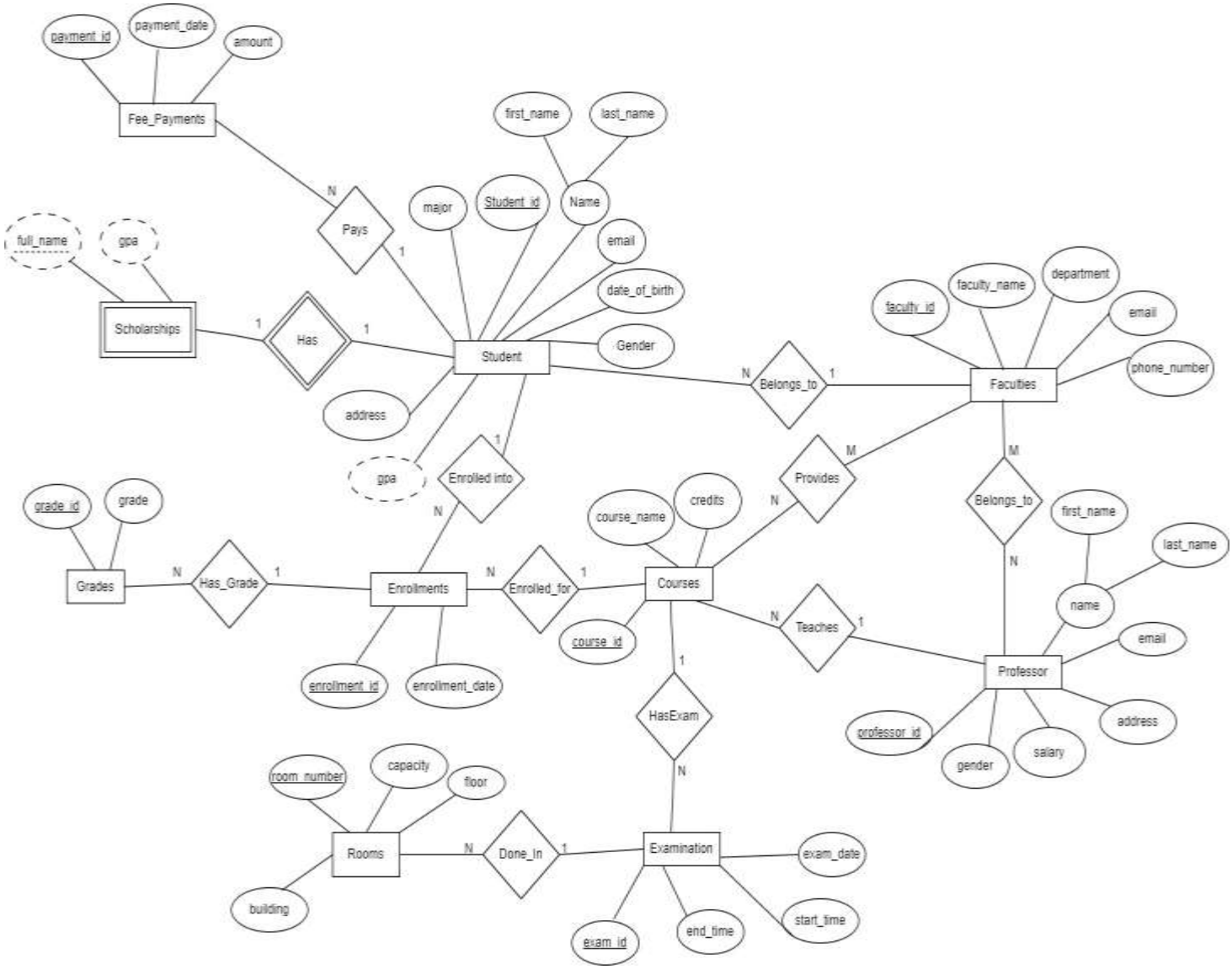
fee_payments

payment_id	student_id	payment_date	amount

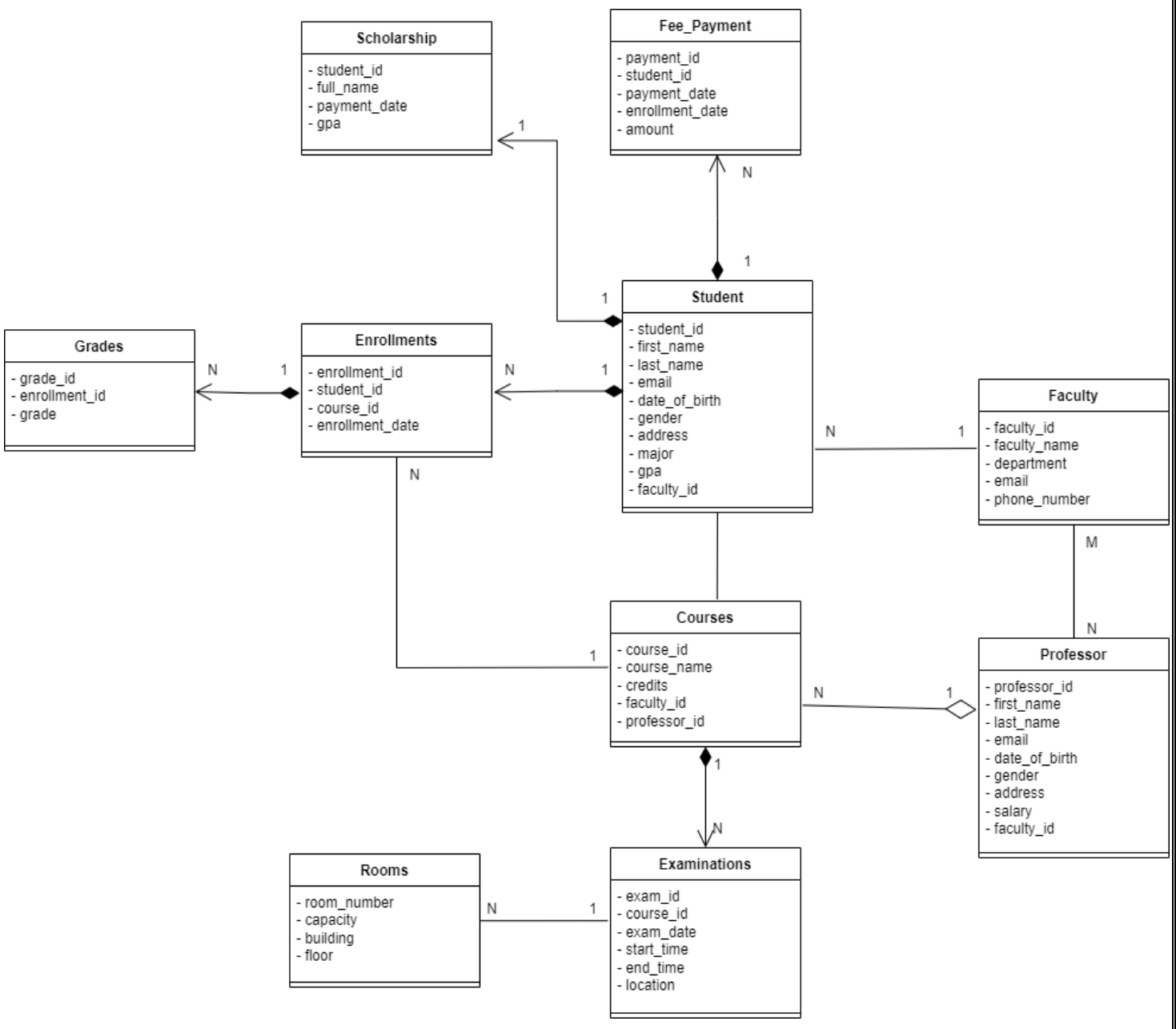
Scholarships

full_name	gpa	student_id

- **ER Diagram:-**


payment_id
payment_date
amount
Fee_Payments
N
Pays
1
first_name
last_name
major
Student_id
Name
email
date_of_birth
Gender
full_name
gpa
Scholarships
1
Has
1
Student
address
gpa
1
Enrolled into
N
N
Belongs_to
1
Faculties
faculty_id
faculty_name
department
email
phone_number
M
Provides
N
M
Belongs_to
N
grade_id
grade
Grades
N
Has_Grade
1
Enrollments
N
Enrolled_for
1
course_name
credits
Courses
course_id
N
Teaches
1
first_name
last_name
name
email
Professor
professor_id
gender
salary
address
enrollment_id
enrollment_date
1
HasExam
N
room_number
capacity
floor
Rooms
N
Done_In
1
Examination
exam_date
start_time
end_time
exam_id
building

- **UML Class Diagram:-**



4) Implementation:-

- First let's show the tables:-

```sql
-- Table for faculties
CREATE TABLE IF NOT EXISTS faculties (
    faculty_id INT AUTO_INCREMENT PRIMARY KEY,
    faculty_name VARCHAR(100),
    department VARCHAR(100),
    email VARCHAR(100),
    phone_number VARCHAR(20)
);

-- Table for students
CREATE TABLE IF NOT EXISTS students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    date_of_birth DATE,
    gender ENUM('Male', 'Female'),
    address VARCHAR(255),
    major VARCHAR(100),
    gpa float,
    faculty_id INT,
    FOREIGN KEY (faculty_id) REFERENCES faculties(faculty_id)
);

-- Table for professors
CREATE TABLE IF NOT EXISTS professors (
    professor_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    date_of_birth DATE,
    gender ENUM('Male', 'Female'),
    address VARCHAR(255),
    salary INT,
    faculty_id INT,
    FOREIGN KEY (faculty_id) REFERENCES faculties(faculty_id)
);
```

```sql
-- Table for courses
CREATE TABLE IF NOT EXISTS courses (
    course_id INT AUTO_INCREMENT PRIMARY KEY,
    course_name VARCHAR(100),
    credits INT,
    faculty_id INT,
    professor_id INT,
    FOREIGN KEY (faculty_id) REFERENCES faculties(faculty_id),
    FOREIGN KEY (professor_id) REFERENCES professors(professor_id)
);

-- Table for enrollments
CREATE TABLE IF NOT EXISTS enrollments (
    enrollment_id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);

-- Table for grades
CREATE TABLE IF NOT EXISTS grades (
    grade_id INT AUTO_INCREMENT PRIMARY KEY,
    enrollment_id INT,
    grade FLOAT,
    FOREIGN KEY (enrollment_id) REFERENCES enrollments(enrollment_id),
    INDEX idx_grade (grade)
);

-- table for rooms
CREATE TABLE IF NOT EXISTS rooms (
    room_number VARCHAR(20) PRIMARY KEY,
    capacity INT,
    building VARCHAR(100),
    floor INT
);
```

```sql
-- Table for examinations
CREATE TABLE IF NOT EXISTS examinations (
    exam_id INT AUTO_INCREMENT PRIMARY KEY,
    course_id INT,
    exam_date DATE,
    start_time TIME,
    end_time TIME,
    location VARCHAR(20),
    FOREIGN KEY (location) REFERENCES rooms(room_number),
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);

-- table for fee payments
CREATE TABLE IF NOT EXISTS fee_payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT,
    payment_date DATE,
    amount DECIMAL(10, 2),
    FOREIGN KEY (student_id) REFERENCES students(student_id)
);


-- Table for scholarships
CREATE TABLE IF NOT EXISTS scholarships (
    student_id INT,
    full_name VARCHAR(100),
    gpa FLOAT,
    FOREIGN KEY (student_id) REFERENCES students(student_id)
);
```

- Then the inserts:-

```sql
-- Inserting sample data into the faculties table
INSERT INTO faculties (faculty_name, department, email, phone_number) VALUES
('Faculty of Computer Science', 'Computer Science', 'cs_faculty@example.com', '123-456-7890'),
('Faculty of Biology', 'Biology', 'bio_faculty@example.com', '987-654-3210');

-- Inserting sample data into the students table
INSERT INTO students (first_name, last_name, email, date_of_birth, gender, address, major, gpa, faculty_i
('John', 'Doe', 'john.doe@example.com', '1998-05-15', 'Male', '123 Main St, City', 'Computer Science', 0.
('Jane', 'Smith', 'jane.smith@example.com', '1997-09-20', 'Female', '456 Elm St, Town', 'Biology', 0.0, 2

-- Inserting sample data into the professors table with faculty_id
INSERT INTO professors (first_name, last_name, email, date_of_birth, gender, address, salary, faculty_id)
('Michael', 'Johnson', 'michael.johnson@example.com', '1975-03-10', 'Male', '789 Oak St Village', 14500,
('Emily', 'Brown', 'emily.brown@example.com', '1982-11-25', 'Female', '101 Pine St, County', 15000, 2);

-- Inserting sample data into the courses table
INSERT INTO courses (course_name, credits, faculty_id, professor_id) VALUES
('Introduction to Computer Science', 3, 1, 1), -- Professor Michael Johnson teaches this course
('Introduction to Biology', 4, 2, 2),          -- Professor Emily Brown teaches this course
('Introduction to Psychology', 3, 2, 2);       -- Professor Emily Brown teaches this course


-- Inserting sample data into the enrollments table
INSERT INTO enrollments (student_id, course_id, enrollment_date) VALUES
(1, 1, '2024-01-15'),
(1, 2, '2024-01-20'),
(2, 1, '2024-01-15'),
(2, 3, '2024-01-25');

-- Inserting sample data into the grades table
INSERT INTO grades (enrollment_id, grade) VALUES
(1, 3.5),
(2, 4.0),
(3, 3.7),
(4, 2.9),
(4, 3.0);
```

```sql
-- Inserting sample data into the rooms table
INSERT INTO rooms (room_number, capacity, building, floor) VALUES
('Room 101', 30, 'Main Building', 1),
('Room 201', 25, 'Science Building', 2),
('Room 102', 35, 'Main Building', 1);

-- Inserting sample data into the examinations table
INSERT INTO examinations (course_id, exam_date, start_time, end_time, location) VALUES
(1, '2024-05-10', '09:00:00', '11:00:00', 'Room 101'),
(2, '2024-05-15', '10:00:00', '12:00:00', 'Room 201'),
(3, '2024-05-20', '09:30:00', '11:30:00', 'Room 102');

-- Inserting sample data into the fee_payments table
INSERT INTO fee_payments (student_id, payment_date, amount) VALUES
(1, '2024-04-01', 500.00),
(2, '2024-04-05', 600.00),
(1, '2024-04-10', 450.00),
(2, '2024-04-15', 550.00);

UPDATE students s
SET s.gpa = (
    SELECT AVG(grade)
    FROM grades g
    WHERE g.enrollment_id IN (SELECT enrollment_id FROM enrollments WHERE student_id = s.student_id)
);

-- Insert students with grades above 3.5 into the scholarships table
INSERT INTO scholarships (student_id, full_name, gpa)
SELECT student_id, CONCAT(first_name, ' ', last_name) AS full_name, gpa
FROM students
WHERE gpa > 3.5;
```

- Here we update the gpa of the student by calculating the average of grades for each enrollment by calculating the average grade for each student
- Insert into scholarship we first take the student_id and full name is first name and last name combined and it only has students with GPA above 3.5

5) Testing:-

- First let's display all the tables

```
185  -- displaying tables
186  select * from faculties;
187  select * from students;
188  select * from grades;
189  select * from professors;
190  select * from enrollments;
191  select * from courses;
192  select * from rooms;
193  select * from examinations;
194  select * from fee_payments;
195  select * from scholarships;
```

- Relational Algebra:-
  - $\pi_{(faculty\_id, faculty\_name, department, email, phone\_number)}(faculties)$
  - $\pi_{(student\_id, first\_name, last\_name, email, date\_of\_birth, gender, address, major, gpa, faculty\_id)}(students)$
  - $\pi_{(grade\_id, enrollment\_id, grade)}(grades)$
  - $\pi_{(professor\_id, first\_name, last\_name, email, date\_of\_birth, gender, address, salary, faculty\_id)}(professors)$
  - $\pi_{(enrollment\_id, student\_id, course\_id, enrollment\_date)}(enrollments)$
  - $\pi_{(course\_id, course\_name, credits, faculty\_id, professor\_id)}(courses)$
  - $\pi_{(room\_number, capacity, building, floor)}(rooms)$
  - $\pi_{(exam\_id, course\_id, exam\_date, start\_time, end\_time, location)}(examinations)$
  - $\pi_{(payment\_id, student\_id, payment\_date, amount)}(fee\_payments)$
  - $\pi_{(student\_id, full\_name, gpa)}(scholarships)$

- Output:

faculty_id	faculty_name	department	email	phone_number
1	Faculty of Computer Science	Computer Science	cs_faculty@example.com	123-456-7890
2	Faculty of Biology	Biology	bio_faculty@example.com	987-654-3210

student_id	first_name	last_name	email	date_of_birth	gender	address	major	gpa	faculty_id
1	John	Doe	john.doe@example.com	1998-05-15	Male	123 Main St, City	Computer Science	3.75	1
2	Jane	Smith	jane.smith@example.com	1997-09-20	Female	456 Elm St, Town	Biology	3.2	2

grade_id	enrollment_id	grade
1	1	3.5
2	2	4
3	3	3.7
4	4	2.9
5	4	3

professor_id	first_name	last_name	email	date_of_birth	gender	address	salary	faculty_id
1	Michael	Johnson	michael.johnson@example.com	1975-03-10	Male	789 Oak St Village	14500	1
2	Emily	Brown	emily.brown@example.com	1982-11-25	Female	101 Pine St, County	15000	2

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-01-15
2	1	2	2024-01-20
3	2	1	2024-01-15
4	2	3	2024-01-25

course_id	course_name	credits	faculty_id	professor_id
1	Introduction to Computer Science	3	1	1
2	Introduction to Biology	4	2	2
3	Introduction to Psychology	3	2	2

room_number	capacity	building	floor
Room 101	30	Main Building	1
Room 102	35	Main Building	1
Room 201	25	Science Building	2

exam_id	course_id	exam_date	start_time	end_time	location
1	1	2024-05-10	09:00:00	11:00:00	Room 101
2	2	2024-05-15	10:00:00	12:00:00	Room 201
3	3	2024-05-20	09:30:00	11:30:00	Room 102

payment_id	student_id	payment_date	amount
1	1	2024-04-01	500.00
2	2	2024-04-05	600.00
3	1	2024-04-10	450.00
4	2	2024-04-15	550.00

student_id	full_name	gpa
1	John Doe	3.75

```sql
-- display only students info that take computer science major
SELECT * FROM students WHERE major = 'Computer Science';

-- display students who are female
SELECT * FROM students WHERE gender = 'Female';

-- display courses info that are in the biology faculty
SELECT * FROM courses WHERE faculty_id = 2;

-- Displaying what courses each student is taking
SELECT
    s.student_id,
    s.first_name,
    s.last_name,
    (
        SELECT course_name
        FROM courses
        WHERE course_id = e.course_id
    ) AS course_name
FROM students s, enrollments e
WHERE s.student_id = e.student_id;
```

- Relational Algebra:-
  - $\sigma_{(major = 'Computer Science')}(students)$
  - $\sigma_{(gender = 'Female')}(students)$
  - $\sigma_{(faculty\_id = 2)}(courses)$
  - $\pi_{(student\_id, first\_name, last\_name, course\_name)} (\sigma_{(students.student\_id = enrollments.student\_id \text{ AND } enrollments.course\_id=courses.course\_id)} (students \bowtie enrollments \bowtie courses))$

- Output:-

student_id	first_name	last_name	email	date_of_birth	gender	address	major	gpa	faculty_id
1	John	Doe	john.doe@example.com	1998-05-15	Male	123 Main St, City	Computer Science	3.75	1

student_id	first_name	last_name	email	date_of_birth	gender	address	major	gpa	faculty_id
2	Jane	Smith	jane.smith@example.com	1997-09-20	Female	456 Elm St, Town	Biology	3.2	2

course_id	course_name	credits	faculty_id	professor_id
2	Introduction to Biology	4	2	2
3	Introduction to Psychology	3	2	2

student_id	first_name	last_name	course_name
1	John	Doe	Introduction to Computer Science
1	John	Doe	Introduction to Biology
2	Jane	Smith	Introduction to Computer Science
2	Jane	Smith	Introduction to Psychology

```sql
-- show students info with total fees between 800 and 1000
SELECT s.student_id, s.first_name, s.last_name, SUM(fp.amount) AS total_fee
FROM students s
LEFT JOIN fee_payments fp ON s.student_id = fp.student_id
GROUP BY s.student_id, s.first_name, s.last_name
HAVING SUM(fp.amount) BETWEEN 800 AND 1000;

-- displays exam and room info for exams that are in rooms on the 1st floor
SELECT e.*, r.*
FROM examinations e
INNER JOIN rooms r ON e.location = r.room_number AND r.floor = 1;

-- Displays students who have GPA > 3.5 and professors with salary > 14500
SELECT
    s.first_name AS student_first_name,
    s.last_name AS student_last_name,
    s.email AS student_email,
    s.gpa AS student_gpa,
    p.first_name AS professor_first_name,
    p.last_name AS professor_last_name,
    p.email AS professor_email,
    p.salary AS professor_salary
FROM students s
INNER JOIN professors p ON s.gpa > 3.5 AND p.salary > 14500;

-- Displays the grades of the students from highest to lowest
SELECT
    s.first_name,
    s.last_name,
    ROUND(g.grade, 2) AS grade
FROM students s
LEFT JOIN enrollments e ON s.student_id = e.student_id
LEFT JOIN grades g ON e.enrollment_id = g.enrollment_id
ORDER BY grade DESC;
```

- Relational Algebra:-
  - $\pi_{(\text{student\_id, first\_name, last\_name, total\_fee})} (\sigma_{(800 < \text{total\_fee} < 1000)} (\pi_{(\text{student\_id, first\_name, last\_name, SUM(amount) AS total\_fee})} (\text{students} \rtimes_{\text{students.student\_id = fee\_payments.student\_id}} \text{fee\_payments}) \gamma (\text{student\_id, first\_name, last\_name}))$
  - $\pi_{(e.*, r.*)} (\sigma_{(r.floor = 1)} (\text{examinations e} \bowtie (\sigma_{(r.floor = 1)} (\text{rooms r}))))$
  - $\pi_{(\text{student\_first\_name, student\_last\_name, student\_email, student\_gpa, professor\_first\_name, professor\_last\_name, professor\_email, professor\_salary})} (\sigma_{(s.gpa > 3.5 \text{ AND } p.salary > 14500)} (\text{students s} \bowtie \text{professors p}))$
  - $\pi_{(\text{first\_name, last\_name, grade})} (\text{students s} \bowtie \text{enrollments e} \bowtie \text{grades g})$ ORDER BY grade DESC

- Output:-

student_id	first_name	last_name	total_fee
1	John	Doe	950.00

exam_id	course_id	exam_date	start_time	end_time	location	room_number	capacity	building	floor
1	1	2024-05-10	09:00:00	11:00:00	Room 101	Room 101	30	Main Building	1
3	3	2024-05-20	09:30:00	11:30:00	Room 102	Room 102	35	Main Building	1

student_first_name	student_last_name	student_email	student_gpa	professor_first_name	professor_last_name	professor_email	professor_salary
John	Doe	john.doe@example.com	3.75	Emily	Brown	emily.brown@example.com	15000

first_name	last_name	grade
John	Doe	4
Jane	Smith	3.7
John	Doe	3.5
Jane	Smith	3
Jane	Smith	2.9