Activity 04: Design a database for Yoobee College

1. Project Scope (Short Paragraph)
   Write a story that defines the purpose and scope of the database. Describe the main entities (e.g., students, lecturers, etc.)
2. Entities and EER Diagram
   List all entities with brief descriptions of their roles and attributes (e.g., Student, Course, Class, Lecturer, etc.)
3. Table Design
   State how many tables are required after mapping the EER to a relational schema?

1. Project Scope (Short Paragraph) –

A. Purpose of the Database

This database is designed to manage core academic operations for an educational institution, specifically tracking:

- Student information and enrollment
- Course offerings and details
- Lecturer assignments and information
- Class scheduling and section management
- Student enrollment in classes with grade tracking

B. Scope of the Database

The database covers the following key academic functions:

a. Student Management

- Stores student personal information
- Tracks academic details
- Maintains a unique identifier for each student

b. Course Catalog

- Defines all available courses with codes, titles, and descriptions
- Tracks credit hours and departmental affiliation
- Supports prerequisite relationships

c. Lecturer Information

- Maintains faculty records with contact and departmental information
- Tracks office locations and employment dates

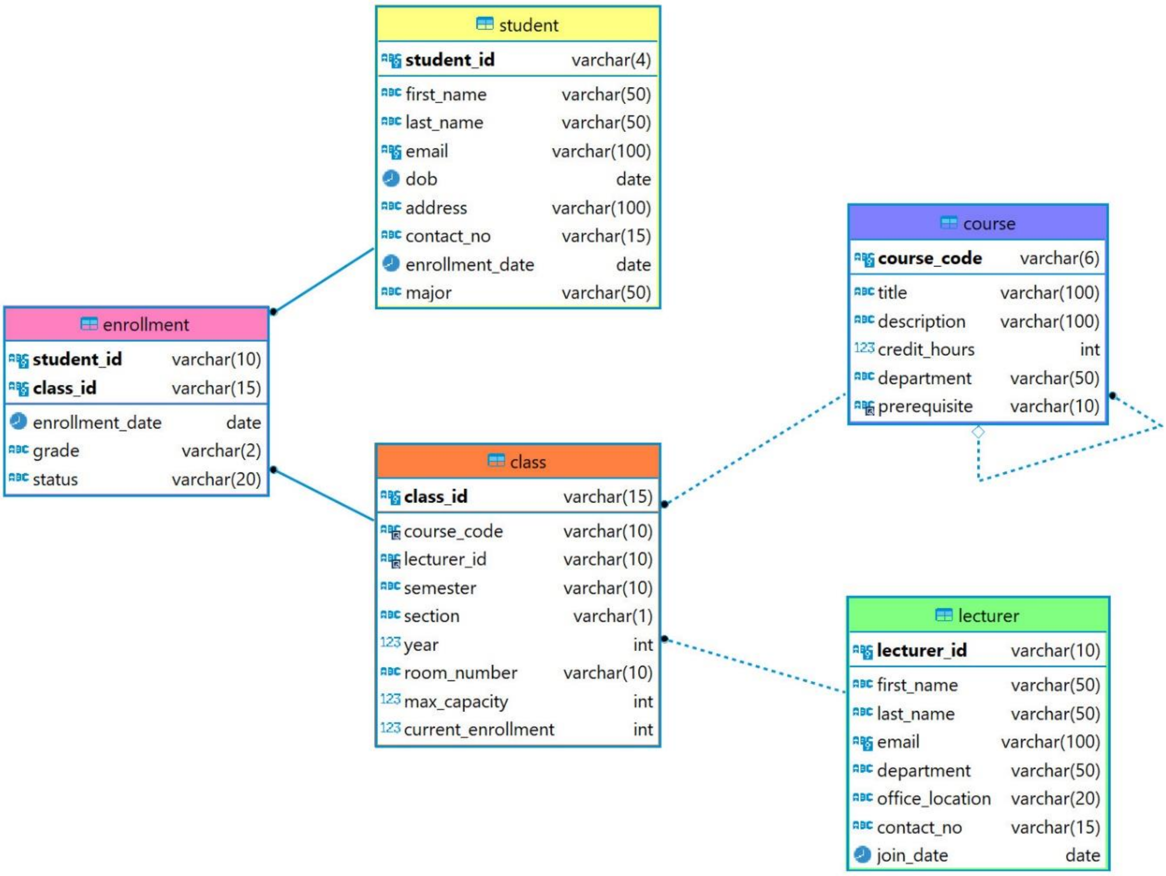
d. Class Scheduling

- Manages specific class instances of courses
- Assigns lecturers to classes
- Organizes by semester/year with section numbering
- Controls room assignments and capacity limits

e. Enrollment System

- Links students to classes they're taking
- Records enrollment dates and status
- Stores final grades for completed classes
- Tracks current enrollment numbers against class capacities

2. Entities and EER Diagram –



3. Table Design –

```
CREATE TABLE student
(
   student_id           VARCHAR(4)     PRIMARY KEY,
   first_name           VARCHAR(50)    NOT NULL,
   last_name            VARCHAR(50)    NOT NULL,
   email                VARCHAR(100)   UNIQUE NOT NULL,
   dob                  DATE,
   address              VARCHAR(100),
   contact_no           VARCHAR(15),
   enrollment_date      DATE           NOT NULL,
   major                VARCHAR(50)
);


CREATE TABLE course
(
   course_code          VARCHAR(6)     PRIMARY KEY,
   title                VARCHAR(100)   NOT NULL,
   description          VARCHAR(100),
   credit_hours         INT            NOT NULL,
   department           VARCHAR(50)    NOT NULL,
   prerequisite         VARCHAR(10),
   --FOREIGN KEY (prerequisite) REFERENCES course (course_code)
);


CREATE TABLE lecturer
(
   lecturer_id          VARCHAR(10)    PRIMARY KEY,
   first_name           VARCHAR(50)    NOT NULL,
   last_name            VARCHAR(50)    NOT NULL,
   email                VARCHAR(100)   UNIQUE NOT NULL,
   department           VARCHAR(50)    NOT NULL,
   office_location      VARCHAR(20),
   contact_no           VARCHAR(15),
   join_date            DATE           NOT NULL
);
```

```
CREATE TABLE class
(
   class_id              VARCHAR(15)    PRIMARY KEY,
   course_code           VARCHAR(10)    NOT NULL,
   lecturer_id           VARCHAR(10)    NOT NULL,
   semester              VARCHAR(10)    NOT NULL,
   section               VARCHAR(1)     NOT NULL,
   year                  INT            NOT NULL,
   room_number           VARCHAR(10),
   max_capacity          INT,
   current_enrollment    INT            DEFAULT 0,
   FOREIGN KEY (course_code) REFERENCES course (course_code),
   FOREIGN KEY (lecturer_id) REFERENCES lecturer (lecturer_id),
   CHECK (current_enrollment <= max_capacity)
);


CREATE TABLE enrollment
(
   student_id            VARCHAR(10)    NOT NULL,
   class_id              VARCHAR(15)    NOT NULL,
   enrollment_date       DATE           NOT NULL,
   grade                 VARCHAR(2),
   status                VARCHAR(20)    DEFAULT 'Enrolled',
   PRIMARY KEY (student_id, class_id),
   FOREIGN KEY (student_id) REFERENCES student (student_id),
   FOREIGN KEY (class_id) REFERENCES class (class_id)
);
```