

**Assignment**

**Course Title:** Database Management System

**Course Code:** CIS222 and CIS222L

**Submitted to:**

Dr. ZZZZZZZZZZZ

Associate Professor

Department of Computing And Information System

Daffodil International University

**Submitted By:**

Name: Nahin Fardosh

ID: 193-16-480

Section: A

Department of Computing And Information System

Daffodil International University

| **Date of Submission: 05/04/2025** |
| --- |

**Theory Part**

**Task 1 -Q1**

The **relational model** was chosen for the DIU student database because of its **simplicity, flexibility, and powerful querying capabilities**. Here's why:

* **Data Integrity & Accuracy**:  
   Relational databases support constraints like **primary keys**, **foreign keys**, and **unique constraints** that maintain data accuracy and relationships between tables .
* **Structured Organization**:  
   Entities such as students, teachers, and courses can be **clearly structured** into tables with defined attributes and relationships, making the data model easy to understand and manage.
* **Efficient Querying & Reporting**:  
   The **SQL language** allows powerful data retrieval using JOIN, GROUP BY, HAVING, etc., which is essential for a university system where analytics and reports are frequently generated.

**Task 1 -Q2**

Mr. Fahim might face several issues while working with the relational model:

* **Normalization Complexity**:  
   Ensuring data is normalized (removing redundancy) without losing important relationships can be challenging and may require multiple iterations of design.
* **Handling Many-to-Many Relationships**:  
   Some relationships ( students enrolling in many courses, teachers teaching many subjects) require **junction tables**. These can become complex to manage.

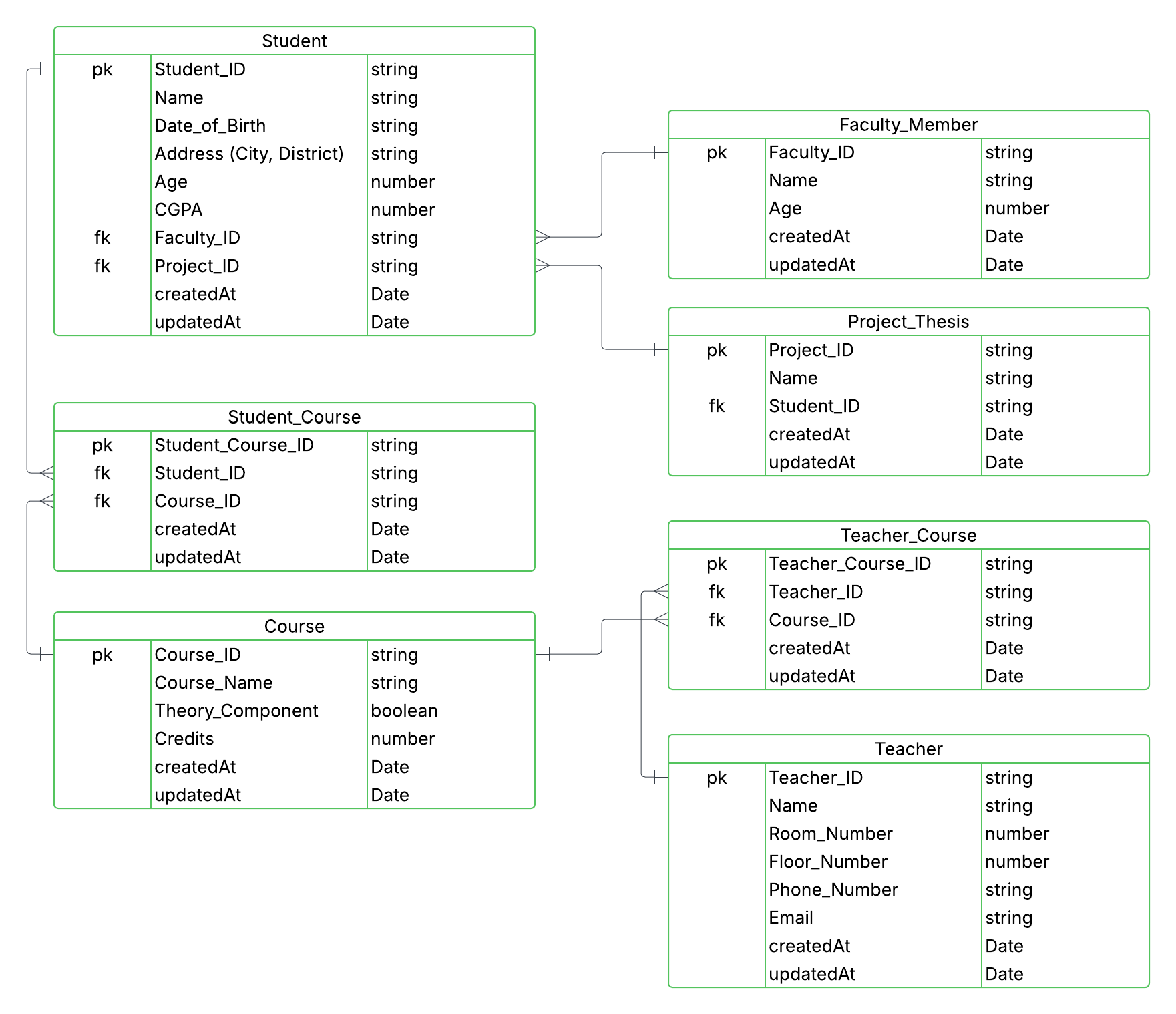
* **Data Redundancy & Anomalies**:  
   Without proper normalization and constraints, there may be issues like **update, delete, and insert anomalies**.
* **Scalability & Performance**:  
   As the student database grows, queries on large tables with multiple joins might **slow down performance** if not optimized properly with indexes and best practices.

**Task 1 -Q3**

Mr. Mehedi most likely described a **Weak Entity** in the blood donation system.

* A **Weak Entity** is one that **cannot exist without being associated** with another entity (known as the identifying or owner entity).
* In a blood donation system, an example could be the **Donation Record**, which cannot exist without a **Donor** entity. The donation details (like date, location, blood type) rely on the existence of the donor.

**Task 2 -Q4**

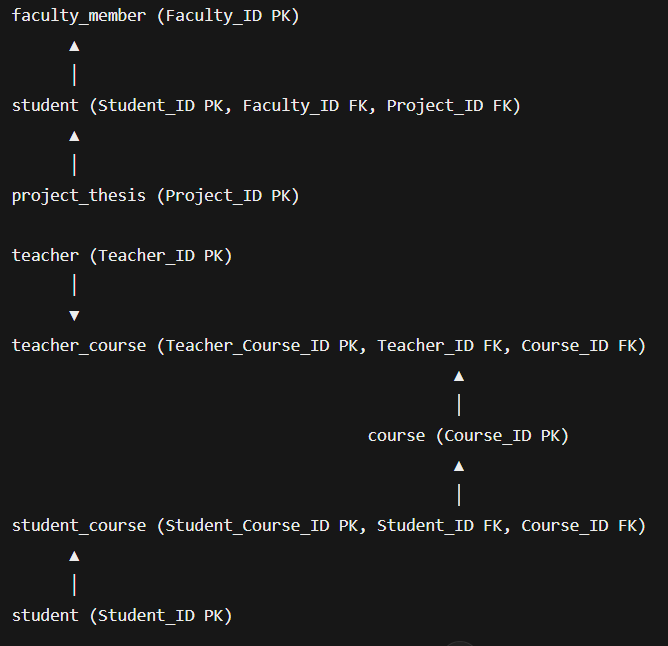


**Task 2 -Q5**

| **Entity Name** | **Attribute Name** | **Data Type** | **Description** | **Key Type** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| faculty\_member | Faculty\_ID | VARCHAR(50) | Unique ID of the faculty | Primary Key |
|  | Name | VARCHAR(100) | Full name of faculty member |  |
|  | Age | INT | Age of the faculty member |  |
|  | createdAt | TIMESTAMP | Creation timestamp |  |
|  | updatedAt | TIMESTAMP | Last update timestamp |  |
| course | Course\_ID | VARCHAR(50) | Unique ID for the course | Primary Key |
|  | Course\_Name | VARCHAR(255) | Name of the course |  |
|  | Theory\_Component | BOOLEAN | Indicates if the course has theory |  |
|  | Credits | INT | Course credit value |  |
|  | createdAt | TIMESTAMP | Creation timestamp |  |
|  | updatedAt | TIMESTAMP | Last update timestamp |  |
| teacher | Teacher\_ID | VARCHAR(50) | Unique ID of the teacher | Primary Key |
|  | Name | VARCHAR(50) | Full name of the teacher |  |
|  | Room\_Number | INT | Room number assigned |  |
|  | Floor\_Number | INT | Floor\_number |  |
|  | Phone\_Number | VARCHAR(20) | Contact number |  |
|  | Email | VARCHAR(100) | Email address |  |
|  | createdAt | TIMESTAMP | Creation timestamp |  |
|  | updatedAt | TIMESTAMP | Last update timestamp |  |
| project\_thesis | Project\_ID | VARCHAR(50) | Unique project ID | Primary Key |
|  | Name | VARCHAR(50) | Name/title of the project |  |
|  | createdAt | TIMESTAMP | Creation timestamp |  |
|  | updatedAt | TIMESTAMP | Last update timestamp |  |
| student | Student\_ID | VARCHAR(50) | Unique student ID | Primary Key |
|  | Name | VARCHAR(100) | Student full name |  |
|  | Date\_of\_Birth | DATE | Date of birth |  |
|  | Address | VARCHAR(255) | Student’s address |  |
|  | Age | INT | Student’s age |  |
|  | CGPA | DECIMAL(3,2) | Student’s CGPA |  |
|  | Faculty\_ID | VARCHAR(50) | Linked faculty ID | Foreign Key |
|  | Project\_ID | VARCHAR(50) | Linked project ID | Foreign Key |
|  | createdAt | TIMESTAMP | Creation timestamp |  |
|  | updatedAt | TIMESTAMP | Last update timestamp |  |
| teacher\_course | Teacher\_Course\_ID | VARCHAR(50) | Unique ID | Primary Key |
|  | Teacher\_ID | VARCHAR(50) | Teacher assigned | Foreign Key |
|  | Course\_ID | VARCHAR(50) | Course assigned | Foreign Key |
|  | createdAt | TIMESTAMP | Creation timestamp |  |
|  | updatedAt | TIMESTAMP | Last update timestamp |  |
| student\_course | Student\_Course\_ID | VARCHAR(50) | Unique ID | Primary Key |
|  | Student\_ID | VARCHAR(50) | Student enrolled | Foreign Key |
|  | Course\_ID | VARCHAR(50) | Enrolled course | Foreign Key |
|  | createdAt | TIMESTAMP | Creation timestamp |  |
|  | updatedAt | TIMESTAMP | Last update timestamp |  |

**Task 2 -Q6**

Below is a textual representation of the relational model (ER-style view):



**Task 2 -Q7**

| **SELECT s.Student\_ID, s.Name, COUNT(sc.Course\_ID) AS Course\_Count**  **FROM student s**  **JOIN student\_course sc ON s.Student\_ID = sc.Student\_ID**  **GROUP BY s.Student\_ID, s.Name**  **HAVING COUNT(sc.Course\_ID) > 2;** |
| --- |

| **SELECT \* FROM teacher**  **WHERE Email LIKE '%diu%'**  **AND Phone\_Number LIKE '017%';** |
| --- |

| **SELECT \* FROM student**  **WHERE Address = 'Dhaka' AND CGPA > 3.65;** |
| --- |

| **SELECT \* FROM student**  **WHERE Address NOT IN ('Dhaka', 'Chittagong');** |
| --- |

| **SELECT s.Student\_ID, s.Name, sc.Course\_ID**  **FROM student s**  **JOIN student\_course sc ON s.Student\_ID = sc.Student\_ID**  **WHERE sc.Course\_ID IN ('CIS222', 'CIS223', 'CIS224');** |
| --- |

| **SELECT \* FROM project\_thesis**  **WHERE Name LIKE '%Library%' OR Name LIKE '%AI%';** |
| --- |

| **SELECT \* FROM student**  **WHERE Age BETWEEN 20 AND 24**  **AND CGPA > 3.71;** |
| --- |

| **SELECT c.Course\_ID, c.Course\_Name**  **FROM course c**  **LEFT JOIN student\_course sc ON c.Course\_ID = sc.Course\_ID**  **WHERE sc.Student\_ID IS NULL;** |
| --- |

| **SELECT c.Course\_ID, c.Course\_Name, COUNT(sc.Student\_ID) AS Student\_Count**  **FROM course c**  **JOIN student\_course sc ON c.Course\_ID = sc.Course\_ID**  **GROUP BY c.Course\_ID, c.Course\_Name**  **HAVING COUNT(sc.Student\_ID) > 1;** |
| --- |

| **SELECT \* FROM student**  **WHERE Faculty\_ID = '710002880';** |
| --- |

**Lab Part**

**Task 1 -QL1**

| -- 1️⃣ Faculty\_Member (Independent)  CREATE TABLE faculty\_member (  Faculty\_ID VARCHAR(50) PRIMARY KEY,  Name VARCHAR(100),  Age INT,  createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  ); |
| --- |

| -- 2️⃣ Course (Independent)  CREATE TABLE course (  Course\_ID VARCHAR(50) PRIMARY KEY,  Course\_Name VARCHAR(255),  Theory\_Component BOOLEAN,  Credits INT,  createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  ); |
| --- |

| -- 3️⃣ Teacher (Independent)  CREATE TABLE teacher (  Teacher\_ID VARCHAR(50) PRIMARY KEY,  Name VARCHAR(100),  Room\_Number INT,  Floor\_Number INT,  Phone\_Number VARCHAR(20),  Email VARCHAR(100),  createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  ); |
| --- |

| -- 4️⃣ Project\_Thesis (Independent, before Student)  CREATE TABLE project\_thesis (  Project\_ID VARCHAR(50) PRIMARY KEY,  Name VARCHAR(255),  createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  ); |
| --- |

| -- 5️⃣ Student (Depends on Faculty\_Member and Project\_Thesis)  CREATE TABLE student (  Student\_ID VARCHAR(50) PRIMARY KEY,  Name VARCHAR(100),  Date\_of\_Birth DATE,  Address VARCHAR(255),  Age INT,  CGPA DECIMAL(3,2),  Faculty\_ID VARCHAR(50),  Project\_ID VARCHAR(50),  createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  FOREIGN KEY (Faculty\_ID) REFERENCES faculty\_member(Faculty\_ID),  FOREIGN KEY (Project\_ID) REFERENCES project\_thesis(Project\_ID)  ); |
| --- |

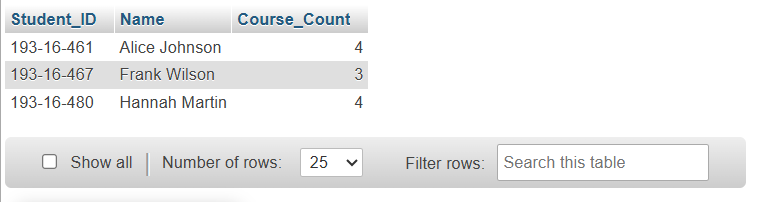
| -- 6️⃣ Teacher\_Course (Depends on Teacher and Course)  CREATE TABLE teacher\_course (  Teacher\_Course\_ID VARCHAR(50) PRIMARY KEY,  Teacher\_ID VARCHAR(50),  Course\_ID VARCHAR(50),  createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  FOREIGN KEY (Teacher\_ID) REFERENCES teacher(Teacher\_ID),  FOREIGN KEY (Course\_ID) REFERENCES course(Course\_ID)  ); |
| --- |

| -- 7️⃣ Student\_Course (Depends on Student and Course)  CREATE TABLE student\_course (  Student\_Course\_ID VARCHAR(50) PRIMARY KEY,  Student\_ID VARCHAR(50),  Course\_ID VARCHAR(50),  createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  FOREIGN KEY (Student\_ID) REFERENCES student(Student\_ID),  FOREIGN KEY (Course\_ID) REFERENCES course(Course\_ID)  ); |
| --- |

**Task 1 -QL2**

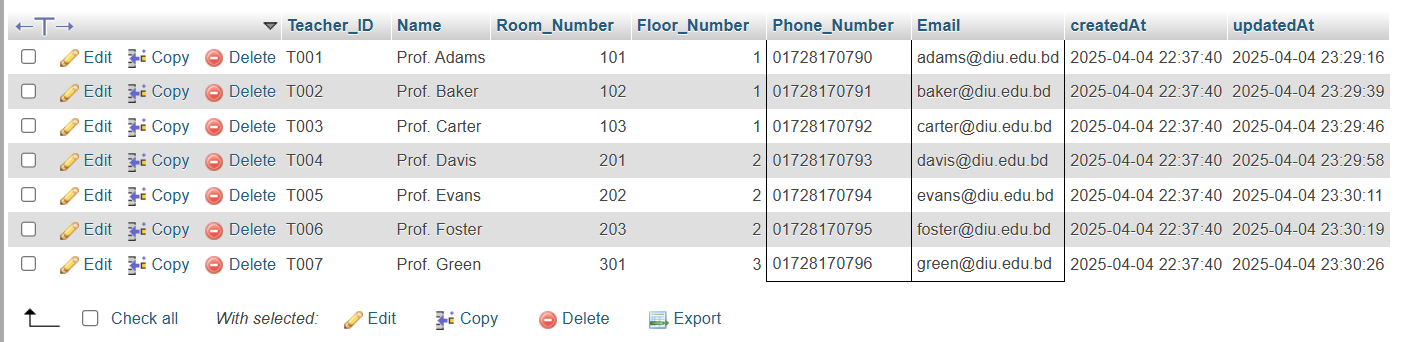
| **SELECT s.Student\_ID, s.Name, COUNT(sc.Course\_ID) AS Course\_Count**  **FROM student s**  **JOIN student\_course sc ON s.Student\_ID = sc.Student\_ID**  **GROUP BY s.Student\_ID, s.Name**  **HAVING COUNT(sc.Course\_ID) > 2;** |
| --- |

**Output:**

****

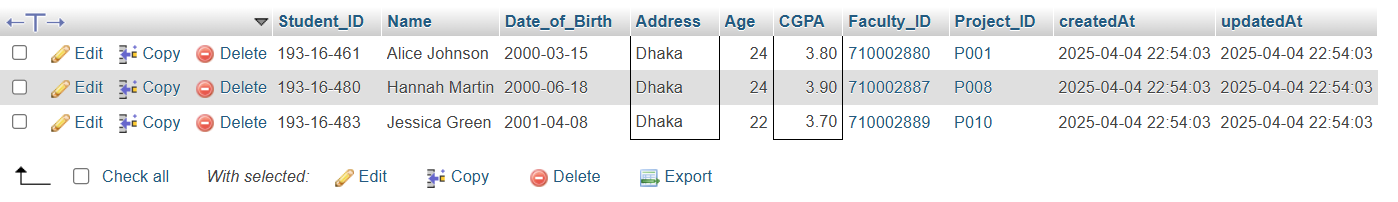
| **SELECT \* FROM teacher**  **WHERE Email LIKE '%diu%'**  **AND Phone\_Number LIKE '017%';** |
| --- |

**Output:**



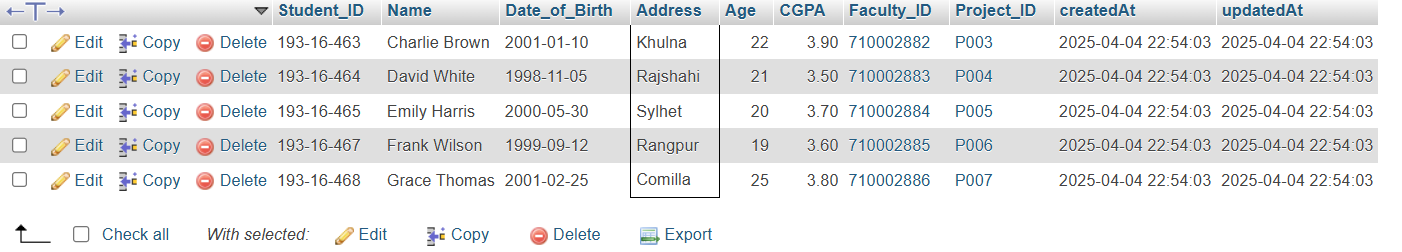
| **SELECT \* FROM student**  **WHERE Address = 'Dhaka' AND CGPA > 3.65;** |
| --- |

**Output:**



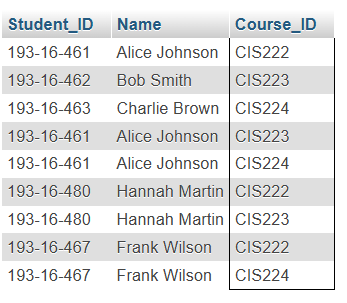
| **SELECT \* FROM student**  **WHERE Address NOT IN ('Dhaka', 'Chittagong');** |
| --- |

**Output:**



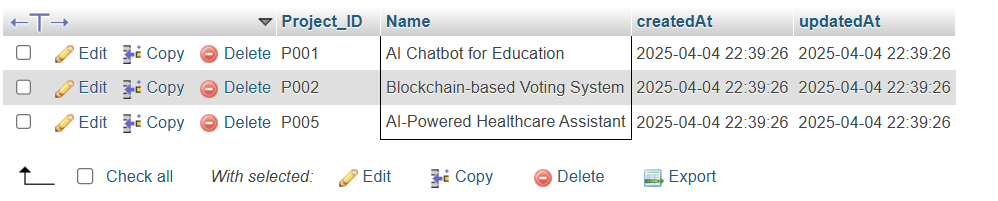
| **SELECT s.Student\_ID, s.Name, sc.Course\_ID**  **FROM student s**  **JOIN student\_course sc ON s.Student\_ID = sc.Student\_ID**  **WHERE sc.Course\_ID IN ('CIS222', 'CIS223', 'CIS224');** |
| --- |

**Output:**



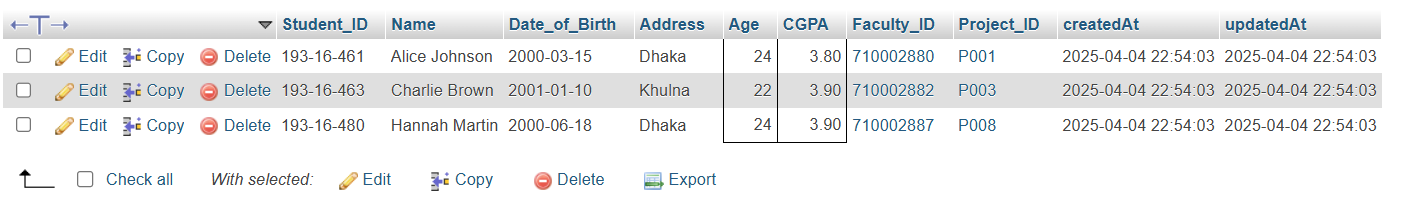
| **SELECT \* FROM project\_thesis**  **WHERE Name LIKE '%Library%' OR Name LIKE '%AI%';** |
| --- |

**Output:**



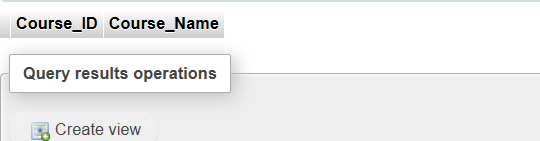
| **SELECT \* FROM student**  **WHERE Age BETWEEN 20 AND 24**  **AND CGPA > 3.71;** |
| --- |

**Output:**

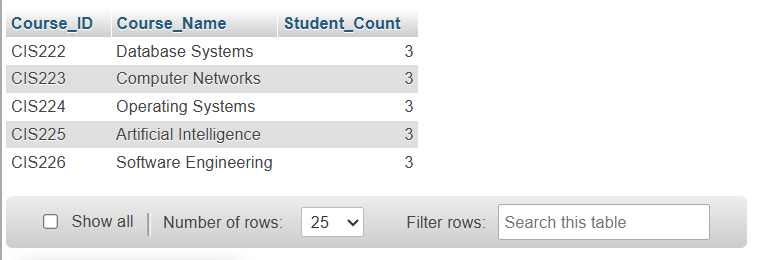


| **SELECT c.Course\_ID, c.Course\_Name**  **FROM course c**  **LEFT JOIN student\_course sc ON c.Course\_ID = sc.Course\_ID**  **WHERE sc.Student\_ID IS NULL;** |
| --- |

**Output :**



| **SELECT c.Course\_ID, c.Course\_Name, COUNT(sc.Student\_ID) AS Student\_Count**  **FROM course c**  **JOIN student\_course sc ON c.Course\_ID = sc.Course\_ID**  **GROUP BY c.Course\_ID, c.Course\_Name**  **HAVING COUNT(sc.Student\_ID) > 1;** |
| --- |

**Output:** 

| **SELECT \* FROM student**  **WHERE Faculty\_ID = '710002880';** |
| --- |

**Output:**



**Task 2 -QL2**

**Video Link :**