`Dhaka International University
Department of CSE

Course Code: 0612-304
Course Title: Database Management System Lab

# Project Proposal on

# Student Result Management System

Submitted by:

**Md. Tamim Hossen**
Roll: 24
**Md. Montasir Hasan ***
Roll: 27
**Nusrat Jahan**
Roll: 38
**Mohammed Shahparan**
Roll: 39
**Nushaiba Kawser Era**
Roll: 41

Submitted to:

**Md. Ayon Mia**
Lecturer,
Department of CSE
Dhaka International University

## Contents

## Overview

It's a centralized digital platform for result management.Teachers can input marks → System calculates grades → Students can view results securely. Built using database design concepts: entities, relationships, and normalization.

Tables: Student, Teacher, Course, Result, ResultDetails, Enrolls, Conducts, etc.

## Objectives

- ❖ Efficiently store and manage students' academic records digitally.
- ❖ Automatically generate accurate result sheets from stored marks.
- ❖ Minimize human errors in result calculation and management.
- ❖ Allow authorized teachers and administrators to input, update, and manage marks.

## Problem Statement

Many educational institutions rely on manual or semi-manual methods to store and process student academic results. This leads to several issues such as delayed result generation, human errors in calculations, duplication of data, and difficulty in access for both staff and students. Manual processing is also labor-intensive, time-consuming, and prone to data loss or unauthorized access, negatively affecting the accuracy and timely delivery of student performance information.

## Solution Statement

The Student Result Management System will automate the collection, storage, and processing of students' academic data. This centralized digital system will enable teachers to input marks easily, automatically calculate final results and grades, and allow students to securely view their results anytime. The system will enhance data accuracy, reduce administrative workload, and provide faster access to academic performance information.

## Entities

✓ **Strong Entities:**

- **Student**

  student_id (PK)
  name
  email
  address
  reg_number
  department
  batch
  roll
  date_of_birth
  age

- **Course**

  course_code (PK)
  course_title
  credit

- **Teacher**

  teacher_id (PK)
  name
  email
  department

- **Result**

  result_id (PK)
  student_id (FK)
  course_code (FK)
  teacher_id (FK)
  marks
  cgpa
  grade

✓ **Weak Entities**

- **ResultDetails**

  result_id (FK)
  marks
  course_code (FK)
  exam_type
  remarks

## ER Diagram (without relationship)

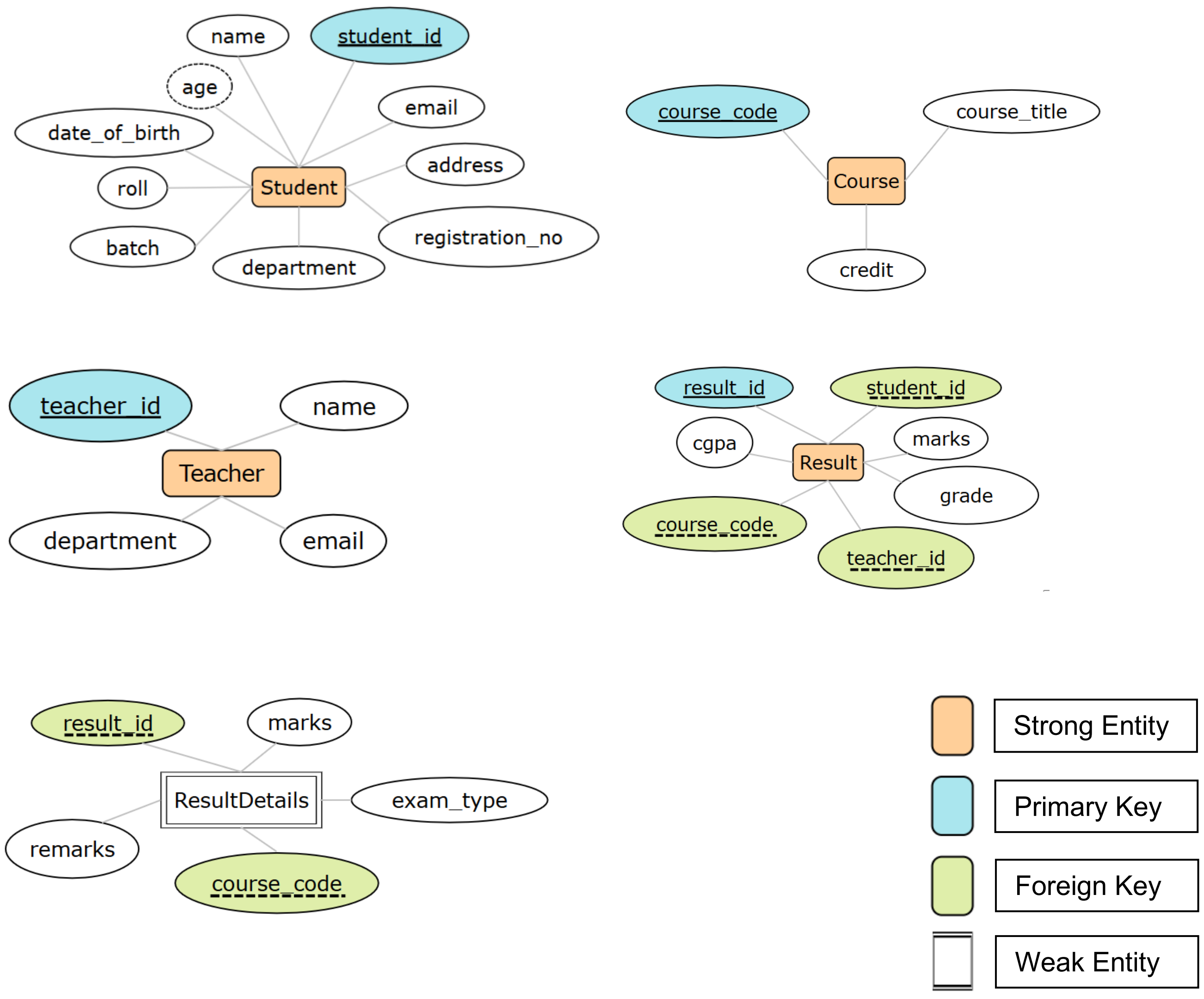


Figure 1: **ER diagram of Student Result Management System without relationship**

## Relationships

- ❖ A student can enroll in many courses, and each course can have many students. → **Many-to-Many (M:N)**
- ❖ One teacher can teach many courses, also each course is taught by many teacher. → **One-to-Many (N:M)**
- ❖ One student can have many results (for different courses). → **One-to-Many (1:M)**
- ❖ One course can have many results (for different students). → **One-to-Many (1:M)**
- ❖ Each result has its detailed information like marks, grade, percentage, etc. → **One-to-One (1:1)** *(if multiple details stored)*

## **ER Diagram** (with relationship)

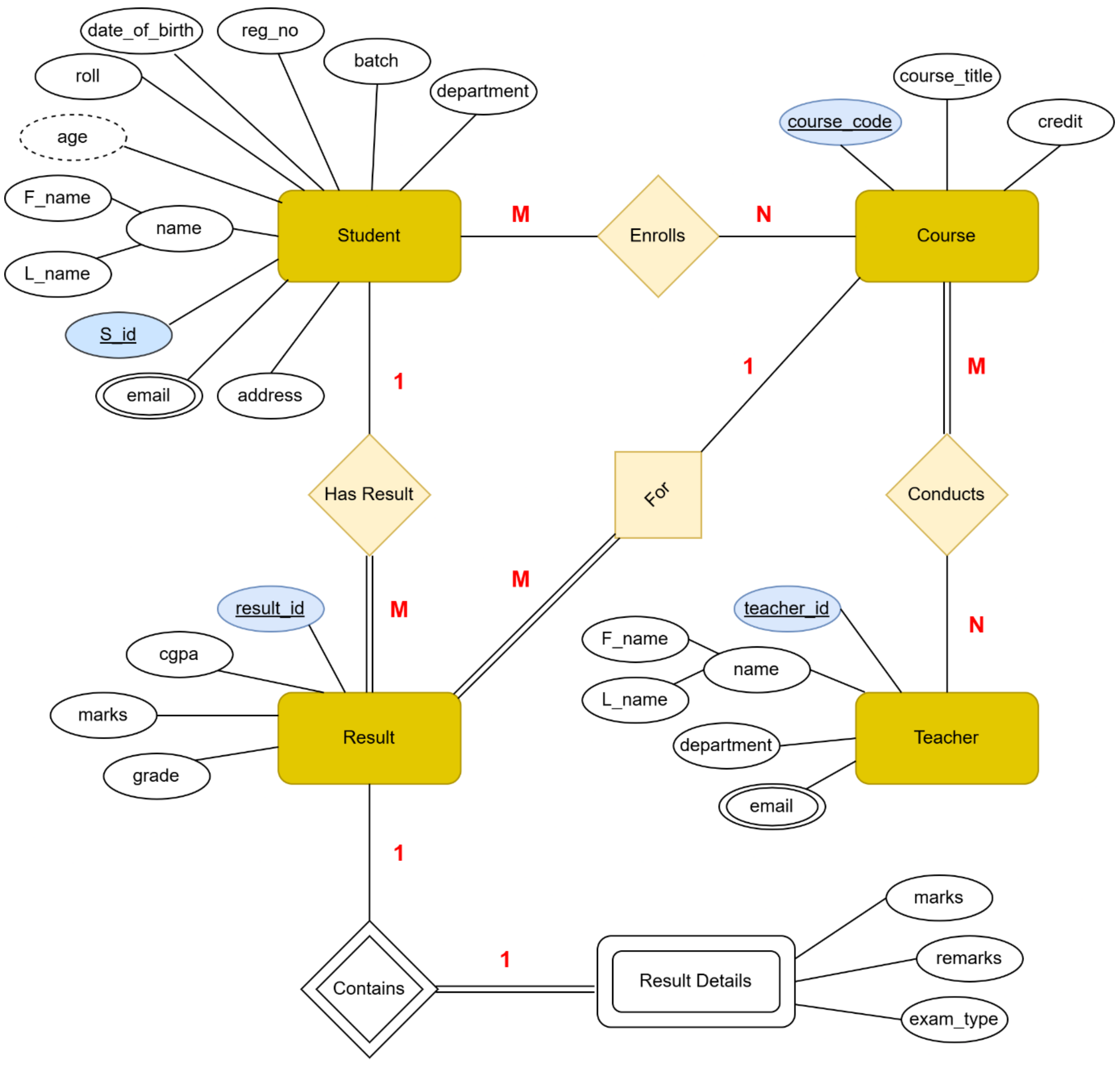


Figure 2: **ER diagram of Student Result Management System with relationship**

## Mapping

**1. Student:**

| Student_id | F_name | L_name | roll | address | Reg_no | DoB | Batch | Departtment |
|---|---|---|---|---|---|---|---|---|

**2. Student_Email:**

| Student_id | S_Email |
|---|---|

**3. Course:**

| Course_id | Course_title | credit | Teacher_id |
|---|---|---|---|

**4. Teacher:**

| Teacher_id | F_name | L_name | Departtment |
|---|---|---|---|

**5. Teacher_Email:**

| Teacher_id | Teacher_Email |
|---|---|

**6. Result:**

| Result_id | cgpa | grade | marks | Course_id | Student_id |
|---|---|---|---|---|---|

**7. Result_Details:**

| Result_id | remarks | Exam_type |
|---|---|---|

**8. Enrolls:**

| Student_id | Course_id |
|---|---|

**9. Conducts:**

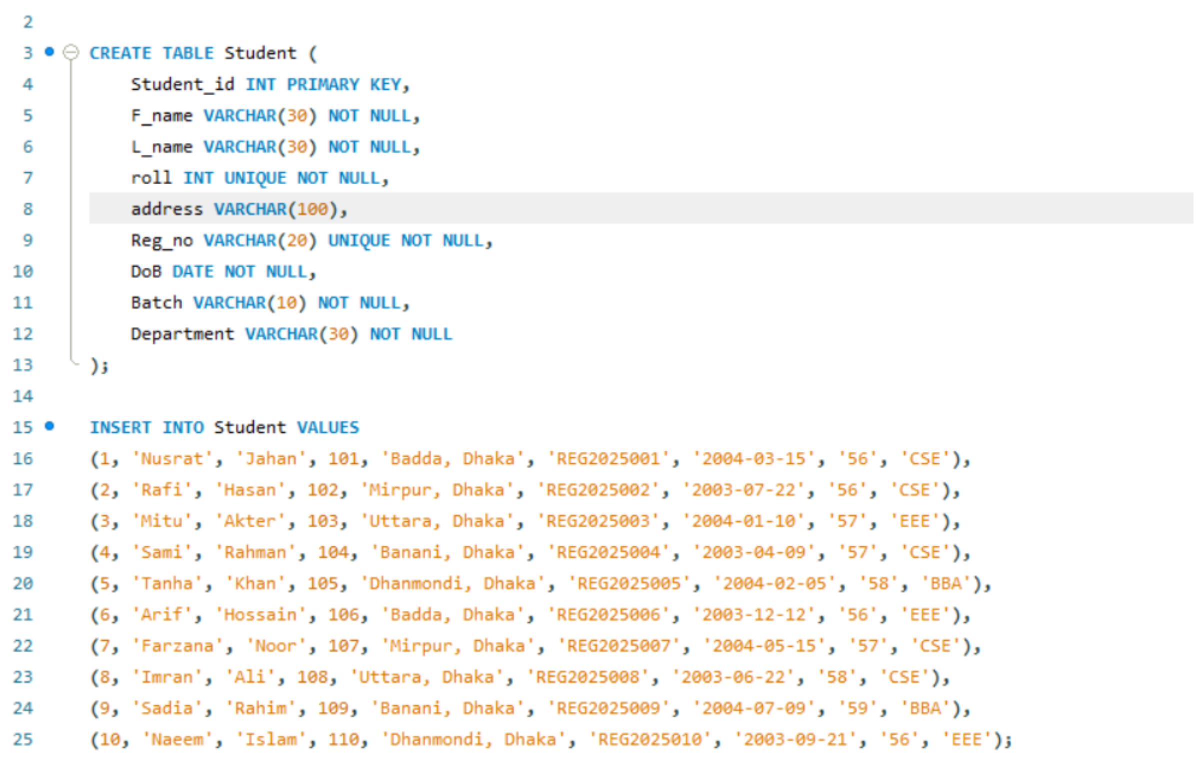
| Course_id | Teacher_id |
|---|---|

## Data Base

### 1. Create Database:

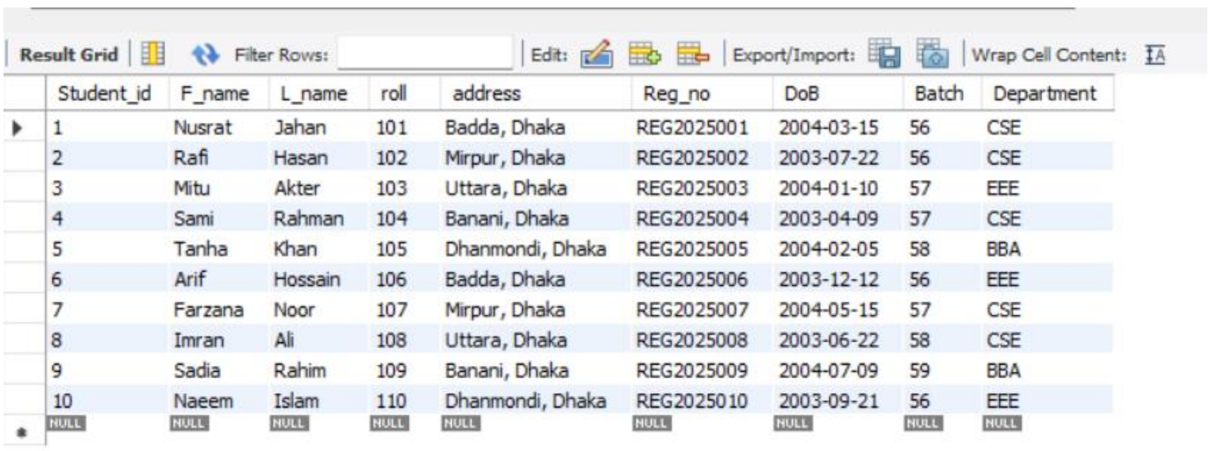
```sql
create Database student_result_management_system_d87;
```

### 2. Student Table:



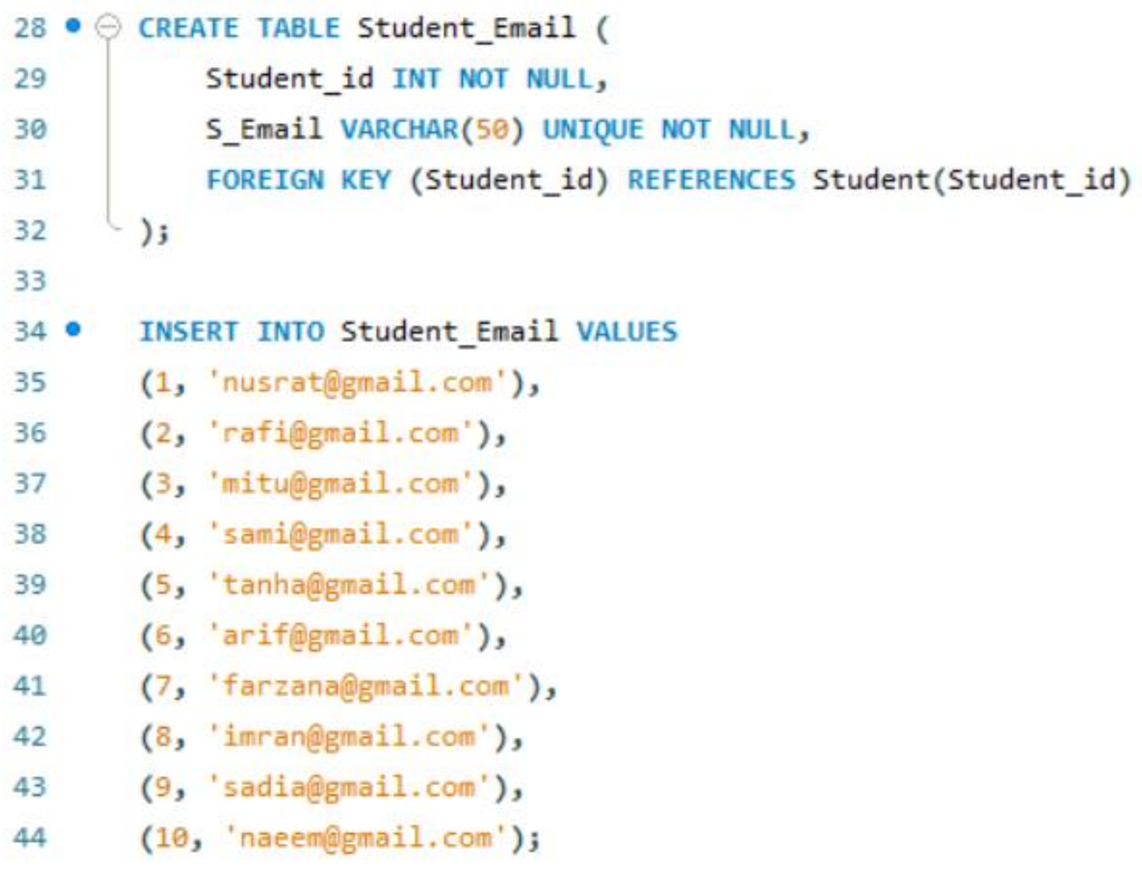
```sql

CREATE TABLE Student (
    Student_id INT PRIMARY KEY,
    F_name VARCHAR(30) NOT NULL,
    L_name VARCHAR(30) NOT NULL,
    roll INT UNIQUE NOT NULL,
    address VARCHAR(100),
    Reg_no VARCHAR(20) UNIQUE NOT NULL,
    DoB DATE NOT NULL,
    Batch VARCHAR(10) NOT NULL,
    Department VARCHAR(30) NOT NULL
);

INSERT INTO Student VALUES
(1, 'Nusrat', 'Jahan', 101, 'Badda, Dhaka', 'REG2025001', '2004-03-15', '56', 'CSE'),
(2, 'Rafi', 'Hasan', 102, 'Mirpur, Dhaka', 'REG2025002', '2003-07-22', '56', 'CSE'),
(3, 'Mitu', 'Akter', 103, 'Uttara, Dhaka', 'REG2025003', '2004-01-10', '57', 'EEE'),
(4, 'Sami', 'Rahman', 104, 'Banani, Dhaka', 'REG2025004', '2003-04-09', '57', 'CSE'),
(5, 'Tanha', 'Khan', 105, 'Dhanmondi, Dhaka', 'REG2025005', '2004-02-05', '58', 'BBA'),
(6, 'Arif', 'Hossain', 106, 'Badda, Dhaka', 'REG2025006', '2003-12-12', '56', 'EEE'),
(7, 'Farzana', 'Noor', 107, 'Mirpur, Dhaka', 'REG2025007', '2004-05-15', '57', 'CSE'),
(8, 'Imran', 'Ali', 108, 'Uttara, Dhaka', 'REG2025008', '2003-06-22', '58', 'CSE'),
(9, 'Sadia', 'Rahim', 109, 'Banani, Dhaka', 'REG2025009', '2004-07-09', '59', 'BBA'),
(10, 'Naeem', 'Islam', 110, 'Dhanmondi, Dhaka', 'REG2025010', '2003-09-21', '56', 'EEE');
```

### Student Table (Output):



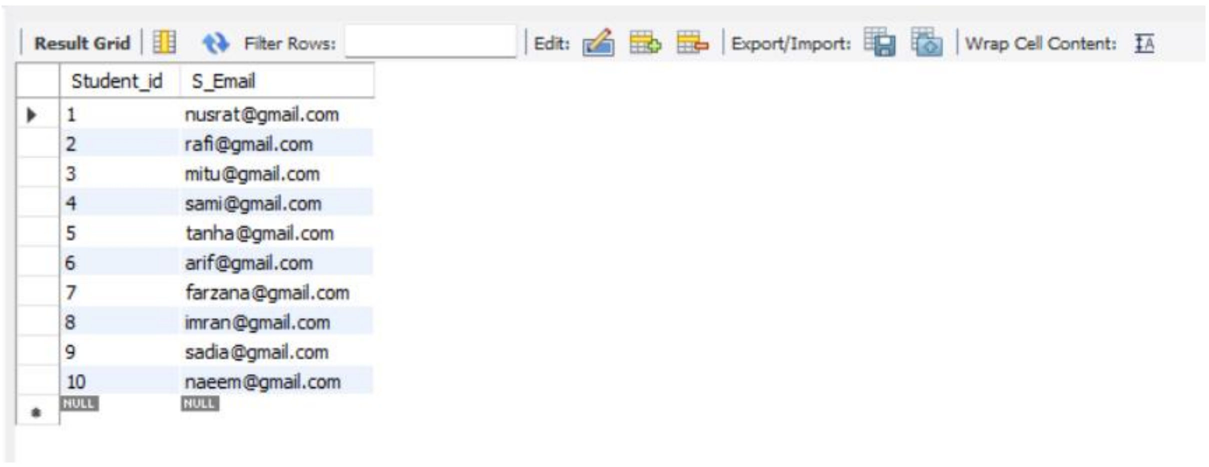
| Student_id | F_name | L_name | roll | address | Reg_no | DoB | Batch | Department |
|---|---|---|---|---|---|---|---|---|
| 1 | Nusrat | Jahan | 101 | Badda, Dhaka | REG2025001 | 2004-03-15 | 56 | CSE |
| 2 | Rafi | Hasan | 102 | Mirpur, Dhaka | REG2025002 | 2003-07-22 | 56 | CSE |
| 3 | Mitu | Akter | 103 | Uttara, Dhaka | REG2025003 | 2004-01-10 | 57 | EEE |
| 4 | Sami | Rahman | 104 | Banani, Dhaka | REG2025004 | 2003-04-09 | 57 | CSE |
| 5 | Tanha | Khan | 105 | Dhanmondi, Dhaka | REG2025005 | 2004-02-05 | 58 | BBA |
| 6 | Arif | Hossain | 106 | Badda, Dhaka | REG2025006 | 2003-12-12 | 56 | EEE |
| 7 | Farzana | Noor | 107 | Mirpur, Dhaka | REG2025007 | 2004-05-15 | 57 | CSE |
| 8 | Imran | Ali | 108 | Uttara, Dhaka | REG2025008 | 2003-06-22 | 58 | CSE |
| 9 | Sadia | Rahim | 109 | Banani, Dhaka | REG2025009 | 2004-07-09 | 59 | BBA |
| 10 | Naeem | Islam | 110 | Dhanmondi, Dhaka | REG2025010 | 2003-09-21 | 56 | EEE |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

3. <u>**Student_Email Table**</u>:



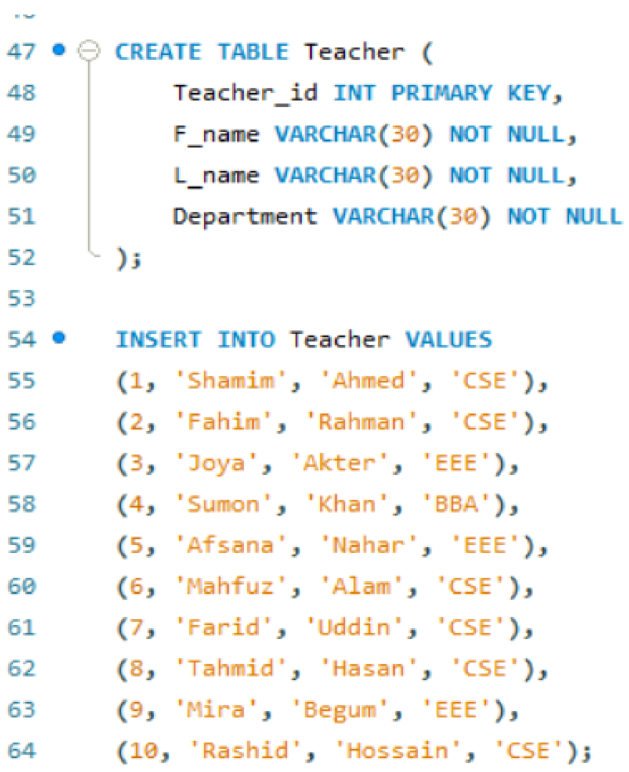
```
CREATE TABLE Student_Email (
    Student_id INT NOT NULL,
    S_Email VARCHAR(50) UNIQUE NOT NULL,
    FOREIGN KEY (Student_id) REFERENCES Student(Student_id)
);

INSERT INTO Student_Email VALUES
(1, 'nusrat@gmail.com'),
(2, 'rafi@gmail.com'),
(3, 'mitu@gmail.com'),
(4, 'sami@gmail.com'),
(5, 'tanha@gmail.com'),
(6, 'arif@gmail.com'),
(7, 'farzana@gmail.com'),
(8, 'imran@gmail.com'),
(9, 'sadia@gmail.com'),
(10, 'naeem@gmail.com');
```

<u>**Student_Email Table (Output)**</u>:



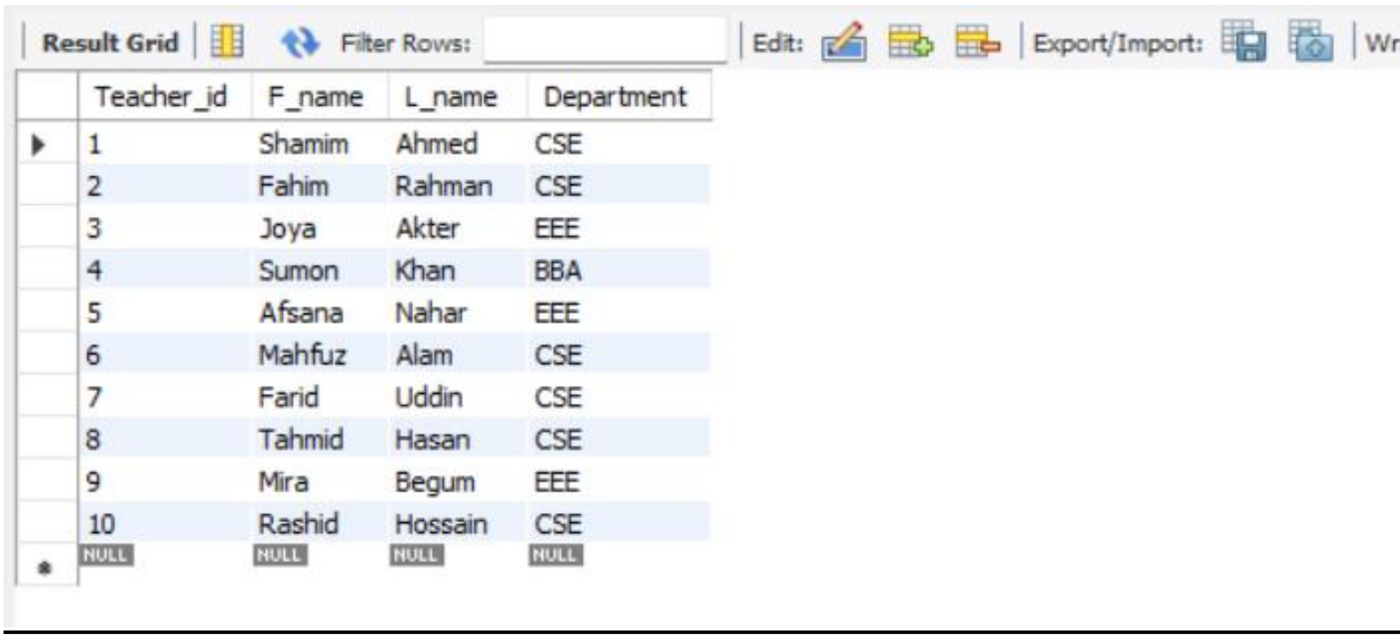
| Student_id | S_Email |
|---|---|
| 1 | nusrat@gmail.com |
| 2 | rafi@gmail.com |
| 3 | mitu@gmail.com |
| 4 | sami@gmail.com |
| 5 | tanha@gmail.com |
| 6 | arif@gmail.com |
| 7 | farzana@gmail.com |
| 8 | imran@gmail.com |
| 9 | sadia@gmail.com |
| 10 | naeem@gmail.com |
| NULL | NULL |

## 4. Teacher Table:



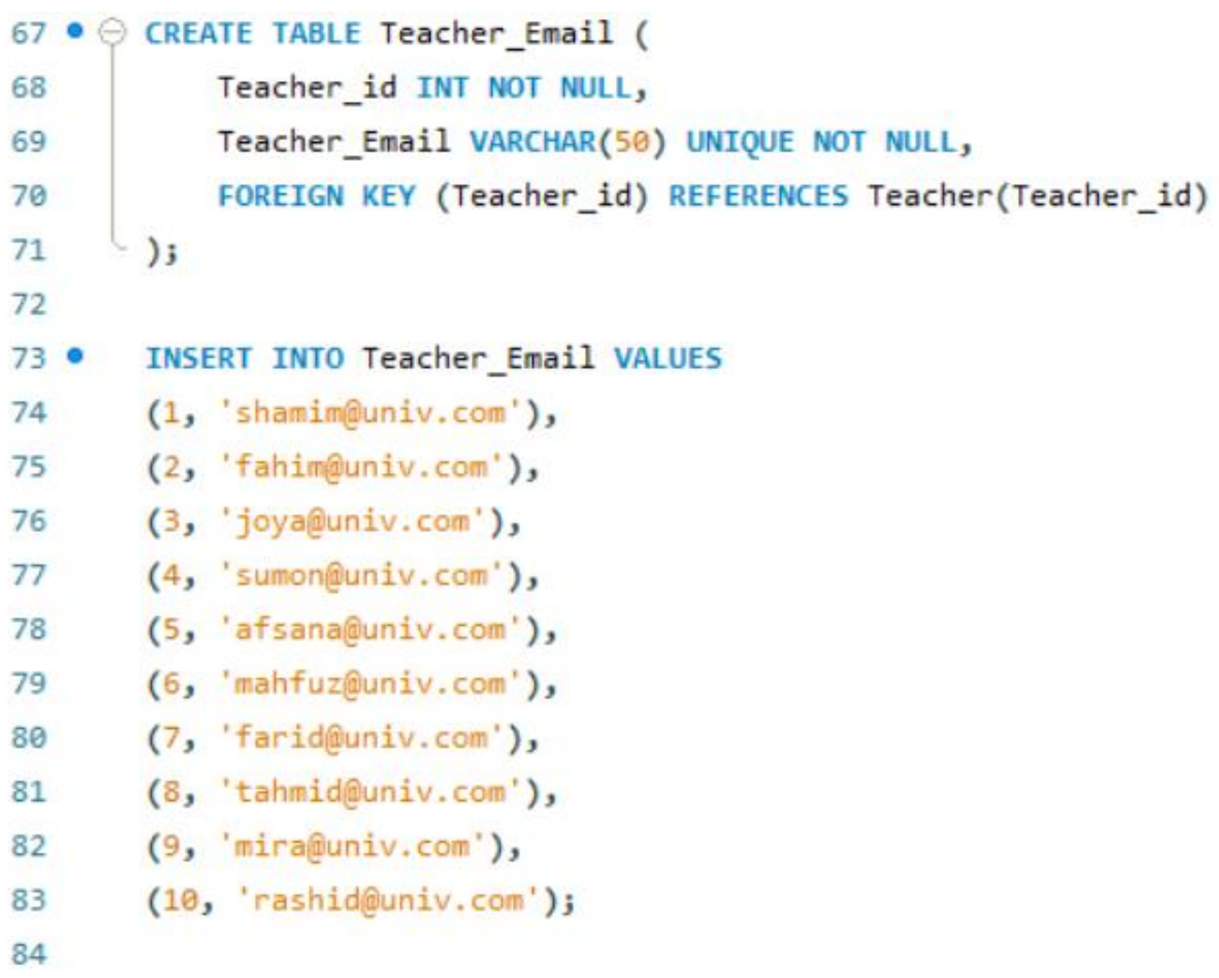
```sql
CREATE TABLE Teacher (
    Teacher_id INT PRIMARY KEY,
    F_name VARCHAR(30) NOT NULL,
    L_name VARCHAR(30) NOT NULL,
    Department VARCHAR(30) NOT NULL
);

INSERT INTO Teacher VALUES
(1, 'Shamim', 'Ahmed', 'CSE'),
(2, 'Fahim', 'Rahman', 'CSE'),
(3, 'Joya', 'Akter', 'EEE'),
(4, 'Sumon', 'Khan', 'BBA'),
(5, 'Afsana', 'Nahar', 'EEE'),
(6, 'Mahfuz', 'Alam', 'CSE'),
(7, 'Farid', 'Uddin', 'CSE'),
(8, 'Tahmid', 'Hasan', 'CSE'),
(9, 'Mira', 'Begum', 'EEE'),
(10, 'Rashid', 'Hossain', 'CSE');
```

## Teacher Table (Output):



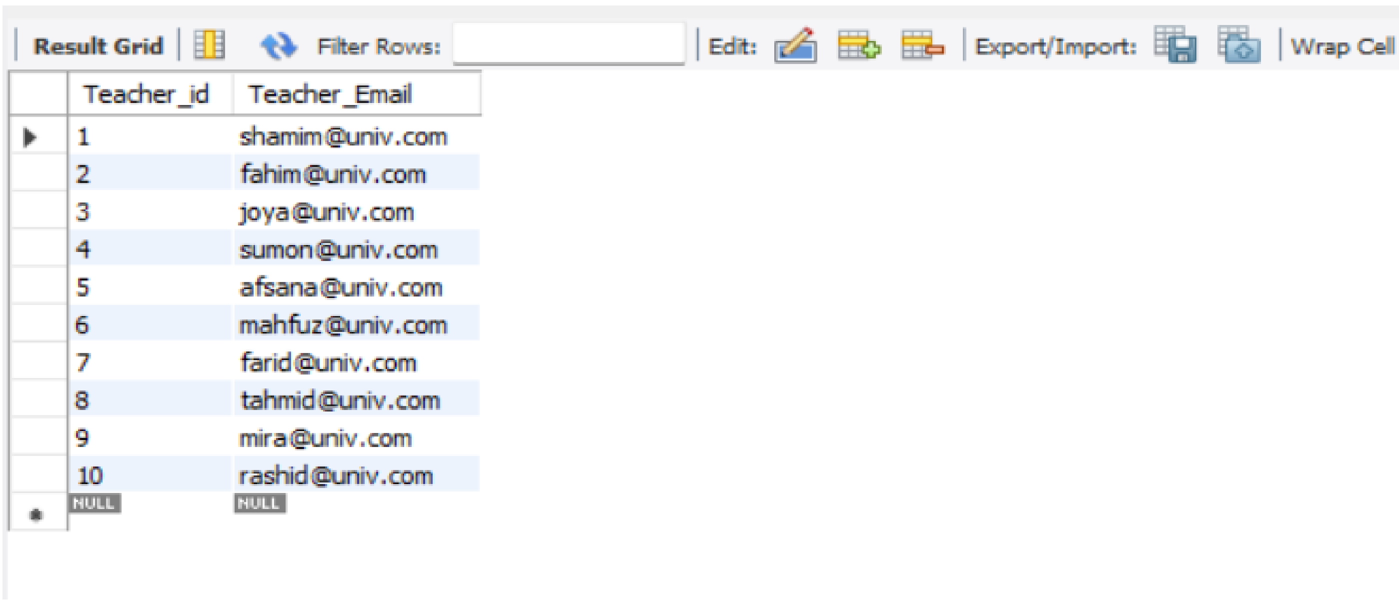
| Teacher_id | F_name | L_name | Department |
|---|---|---|---|
| 1 | Shamim | Ahmed | CSE |
| 2 | Fahim | Rahman | CSE |
| 3 | Joya | Akter | EEE |
| 4 | Sumon | Khan | BBA |
| 5 | Afsana | Nahar | EEE |
| 6 | Mahfuz | Alam | CSE |
| 7 | Farid | Uddin | CSE |
| 8 | Tahmid | Hasan | CSE |
| 9 | Mira | Begum | EEE |
| 10 | Rashid | Hossain | CSE |
| NULL | NULL | NULL | NULL |

5. **<u>Teacher_Email Table:</u>**



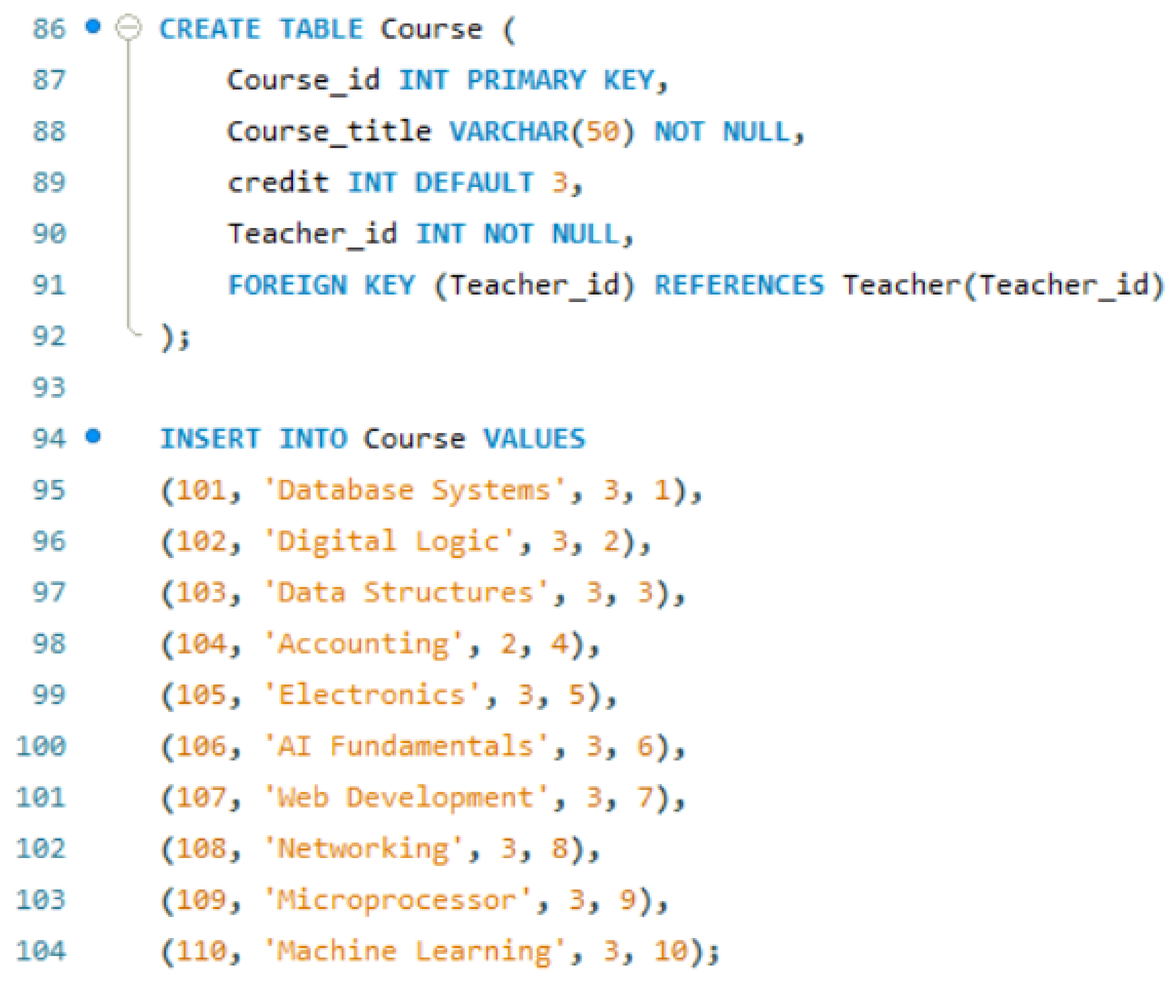
```sql
CREATE TABLE Teacher_Email (
    Teacher_id INT NOT NULL,
    Teacher_Email VARCHAR(50) UNIQUE NOT NULL,
    FOREIGN KEY (Teacher_id) REFERENCES Teacher(Teacher_id)
);

INSERT INTO Teacher_Email VALUES
(1, 'shamim@univ.com'),
(2, 'fahim@univ.com'),
(3, 'joya@univ.com'),
(4, 'sumon@univ.com'),
(5, 'afsana@univ.com'),
(6, 'mahfuz@univ.com'),
(7, 'farid@univ.com'),
(8, 'tahmid@univ.com'),
(9, 'mira@univ.com'),
(10, 'rashid@univ.com');
```

**<u>Teacher_Email Table (Output):</u>**



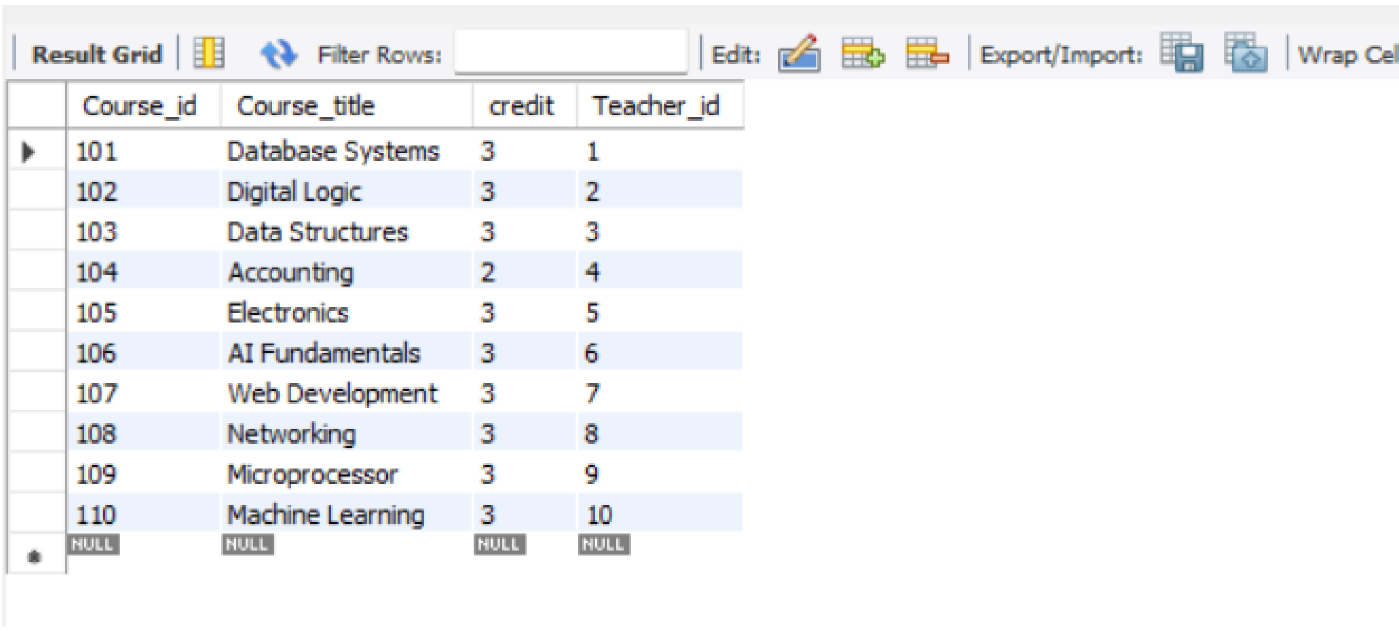
| Teacher_id | Teacher_Email |
|---|---|
| 1 | shamim@univ.com |
| 2 | fahim@univ.com |
| 3 | joya@univ.com |
| 4 | sumon@univ.com |
| 5 | afsana@univ.com |
| 6 | mahfuz@univ.com |
| 7 | farid@univ.com |
| 8 | tahmid@univ.com |
| 9 | mira@univ.com |
| 10 | rashid@univ.com |
| NULL | NULL |

## 6. Course Table:



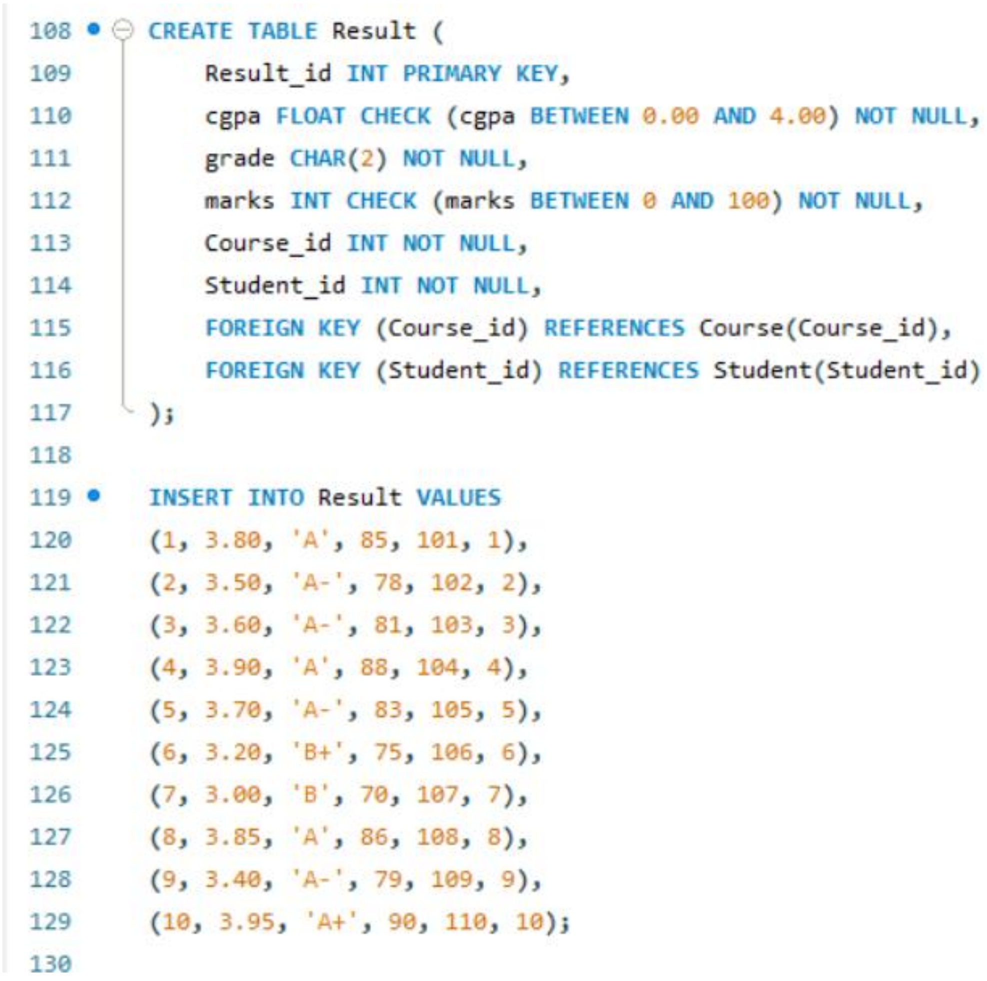
```
CREATE TABLE Course (
    Course_id INT PRIMARY KEY,
    Course_title VARCHAR(50) NOT NULL,
    credit INT DEFAULT 3,
    Teacher_id INT NOT NULL,
    FOREIGN KEY (Teacher_id) REFERENCES Teacher(Teacher_id)
);

INSERT INTO Course VALUES
(101, 'Database Systems', 3, 1),
(102, 'Digital Logic', 3, 2),
(103, 'Data Structures', 3, 3),
(104, 'Accounting', 2, 4),
(105, 'Electronics', 3, 5),
(106, 'AI Fundamentals', 3, 6),
(107, 'Web Development', 3, 7),
(108, 'Networking', 3, 8),
(109, 'Microprocessor', 3, 9),
(110, 'Machine Learning', 3, 10);
```

## Course Table (Output):



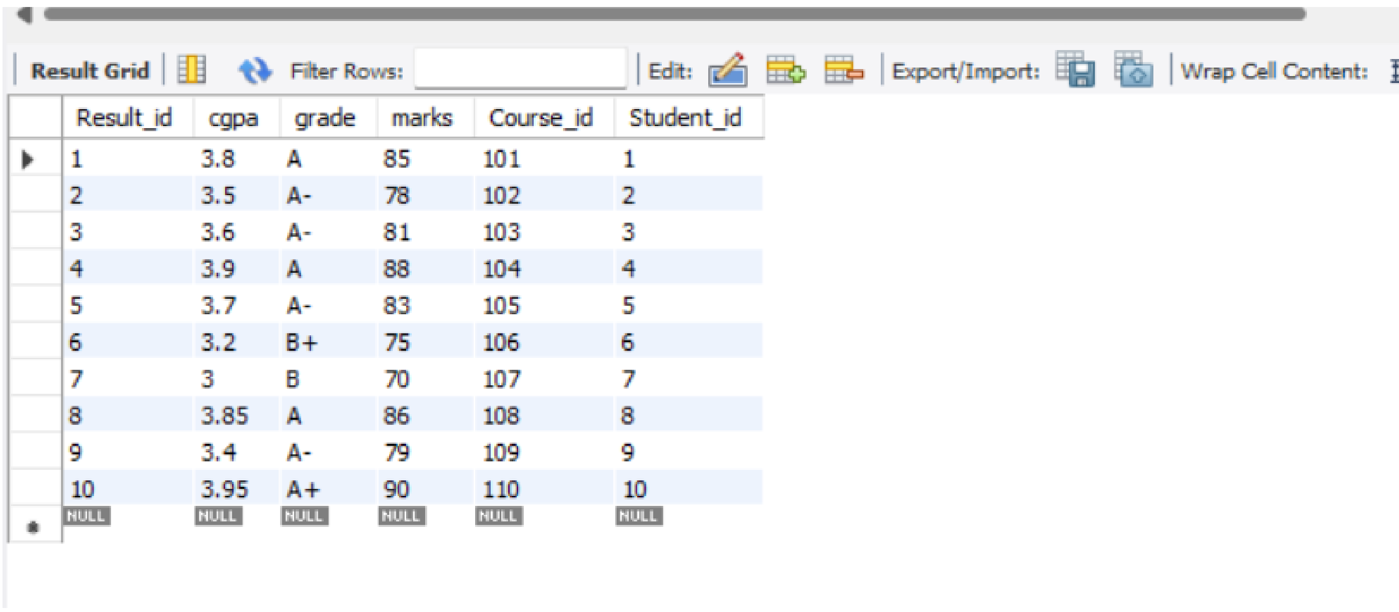
| Course_id | Course_title | credit | Teacher_id |
|---|---|---|---|
| 101 | Database Systems | 3 | 1 |
| 102 | Digital Logic | 3 | 2 |
| 103 | Data Structures | 3 | 3 |
| 104 | Accounting | 2 | 4 |
| 105 | Electronics | 3 | 5 |
| 106 | AI Fundamentals | 3 | 6 |
| 107 | Web Development | 3 | 7 |
| 108 | Networking | 3 | 8 |
| 109 | Microprocessor | 3 | 9 |
| 110 | Machine Learning | 3 | 10 |
| NULL | NULL | NULL | NULL |

## 7. Result Table:



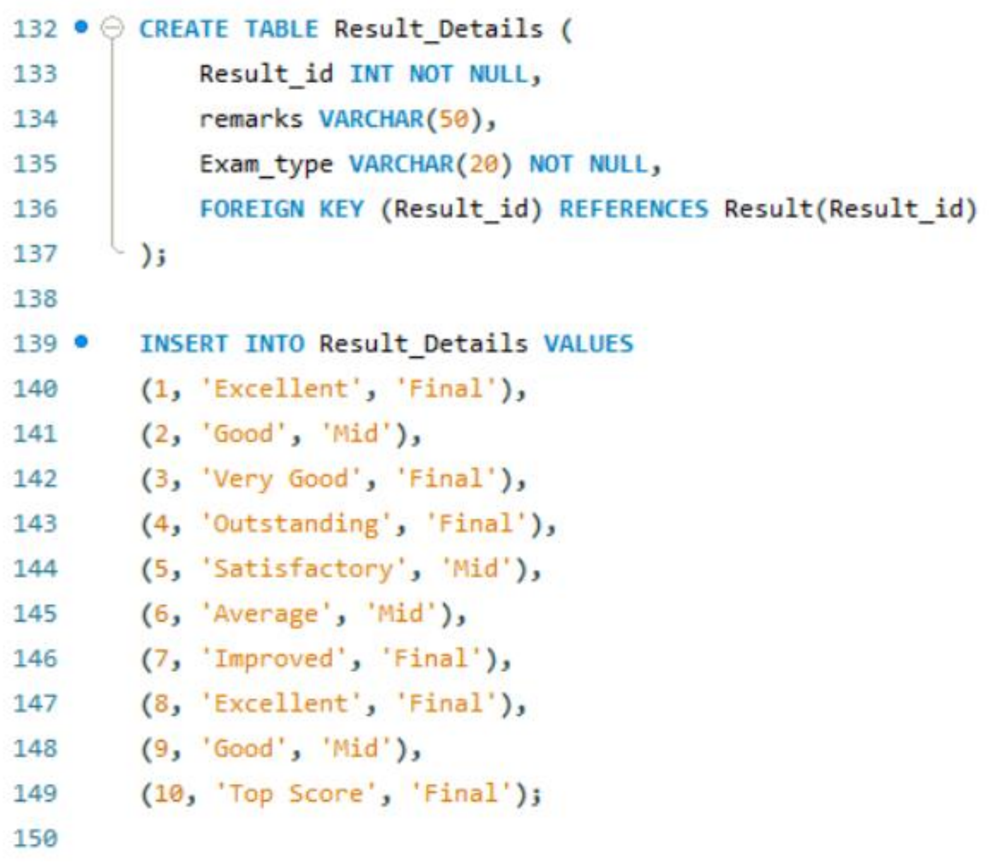
```
CREATE TABLE Result (
    Result_id INT PRIMARY KEY,
    cgpa FLOAT CHECK (cgpa BETWEEN 0.00 AND 4.00) NOT NULL,
    grade CHAR(2) NOT NULL,
    marks INT CHECK (marks BETWEEN 0 AND 100) NOT NULL,
    Course_id INT NOT NULL,
    Student_id INT NOT NULL,
    FOREIGN KEY (Course_id) REFERENCES Course(Course_id),
    FOREIGN KEY (Student_id) REFERENCES Student(Student_id)
);

INSERT INTO Result VALUES
(1, 3.80, 'A', 85, 101, 1),
(2, 3.50, 'A-', 78, 102, 2),
(3, 3.60, 'A-', 81, 103, 3),
(4, 3.90, 'A', 88, 104, 4),
(5, 3.70, 'A-', 83, 105, 5),
(6, 3.20, 'B+', 75, 106, 6),
(7, 3.00, 'B', 70, 107, 7),
(8, 3.85, 'A', 86, 108, 8),
(9, 3.40, 'A-', 79, 109, 9),
(10, 3.95, 'A+', 90, 110, 10);
```

## Result Table (Output):



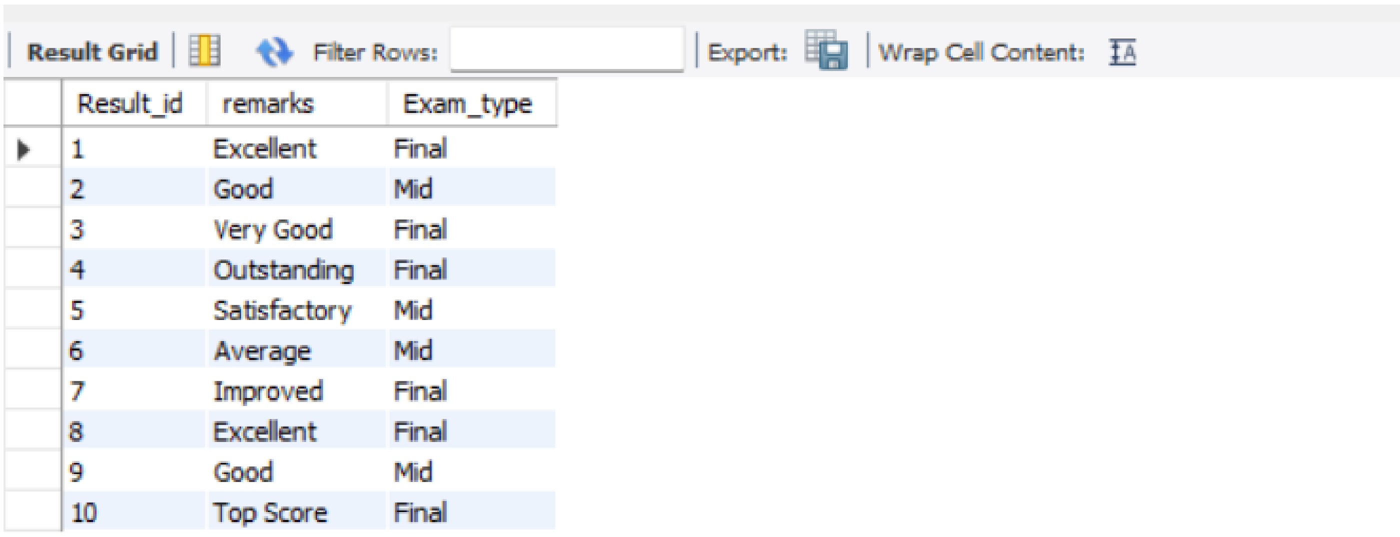
| Result_id | cgpa | grade | marks | Course_id | Student_id |
|---|---|---|---|---|---|
| 1 | 3.8 | A | 85 | 101 | 1 |
| 2 | 3.5 | A- | 78 | 102 | 2 |
| 3 | 3.6 | A- | 81 | 103 | 3 |
| 4 | 3.9 | A | 88 | 104 | 4 |
| 5 | 3.7 | A- | 83 | 105 | 5 |
| 6 | 3.2 | B+ | 75 | 106 | 6 |
| 7 | 3 | B | 70 | 107 | 7 |
| 8 | 3.85 | A | 86 | 108 | 8 |
| 9 | 3.4 | A- | 79 | 109 | 9 |
| 10 | 3.95 | A+ | 90 | 110 | 10 |
| NULL | NULL | NULL | NULL | NULL | NULL |

8. **Result_Details Table**:



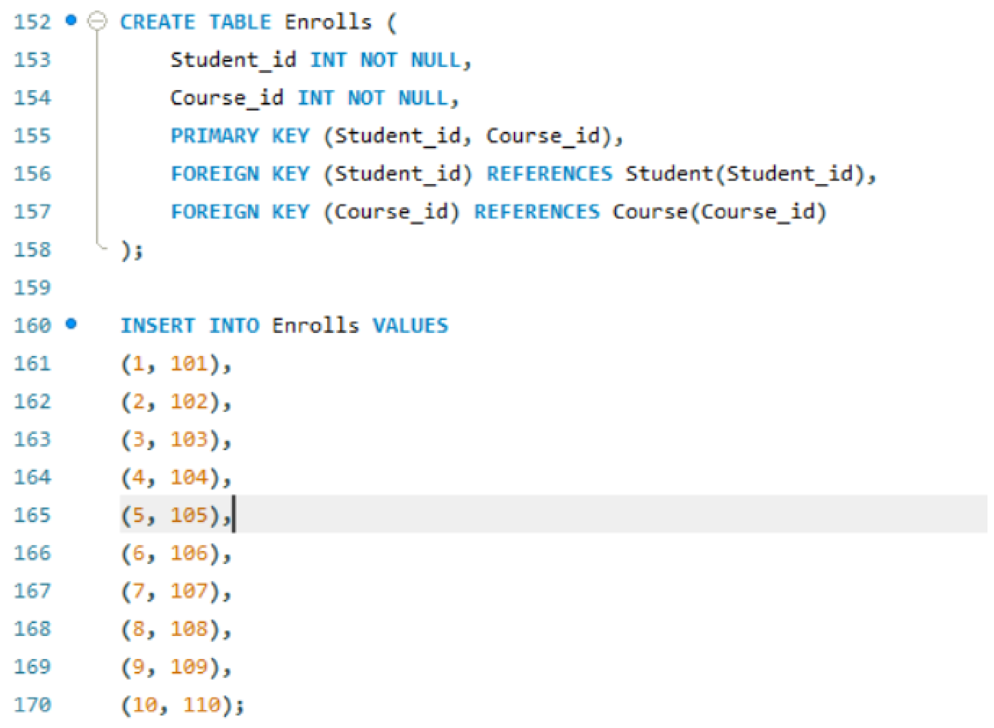
```sql
CREATE TABLE Result_Details (
    Result_id INT NOT NULL,
    remarks VARCHAR(50),
    Exam_type VARCHAR(20) NOT NULL,
    FOREIGN KEY (Result_id) REFERENCES Result(Result_id)
);

INSERT INTO Result_Details VALUES
(1, 'Excellent', 'Final'),
(2, 'Good', 'Mid'),
(3, 'Very Good', 'Final'),
(4, 'Outstanding', 'Final'),
(5, 'Satisfactory', 'Mid'),
(6, 'Average', 'Mid'),
(7, 'Improved', 'Final'),
(8, 'Excellent', 'Final'),
(9, 'Good', 'Mid'),
(10, 'Top Score', 'Final');
```

**Result_Details Table (Output)**:



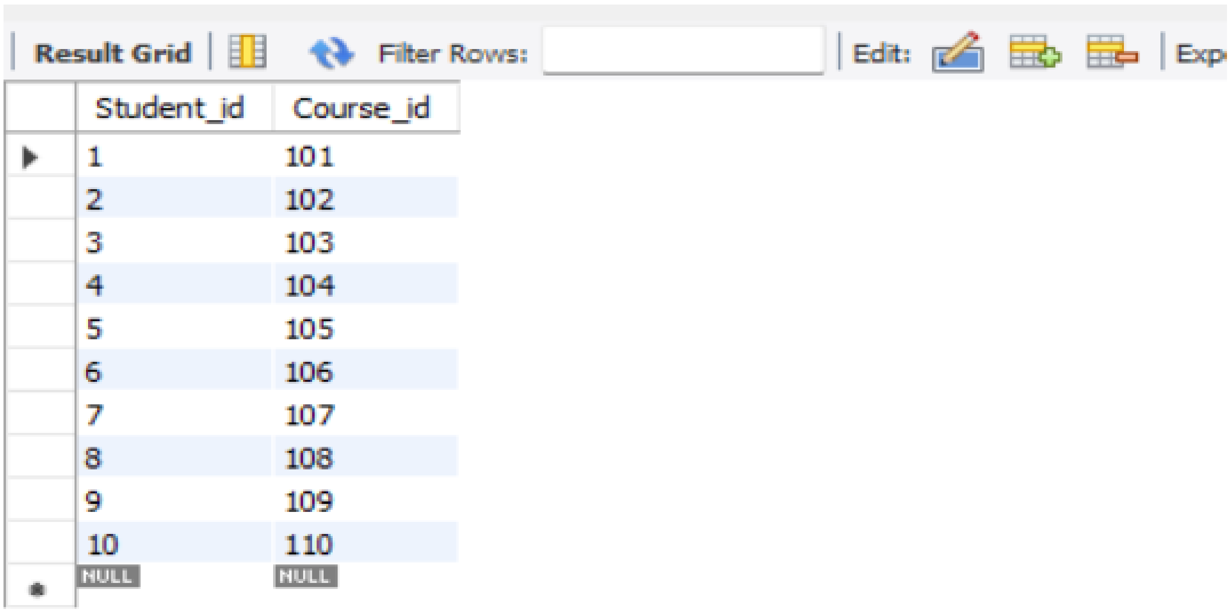
| Result_id | remarks | Exam_type |
|---|---|---|
| 1 | Excellent | Final |
| 2 | Good | Mid |
| 3 | Very Good | Final |
| 4 | Outstanding | Final |
| 5 | Satisfactory | Mid |
| 6 | Average | Mid |
| 7 | Improved | Final |
| 8 | Excellent | Final |
| 9 | Good | Mid |
| 10 | Top Score | Final |

**9. <u>Enrolls Table</u>:**



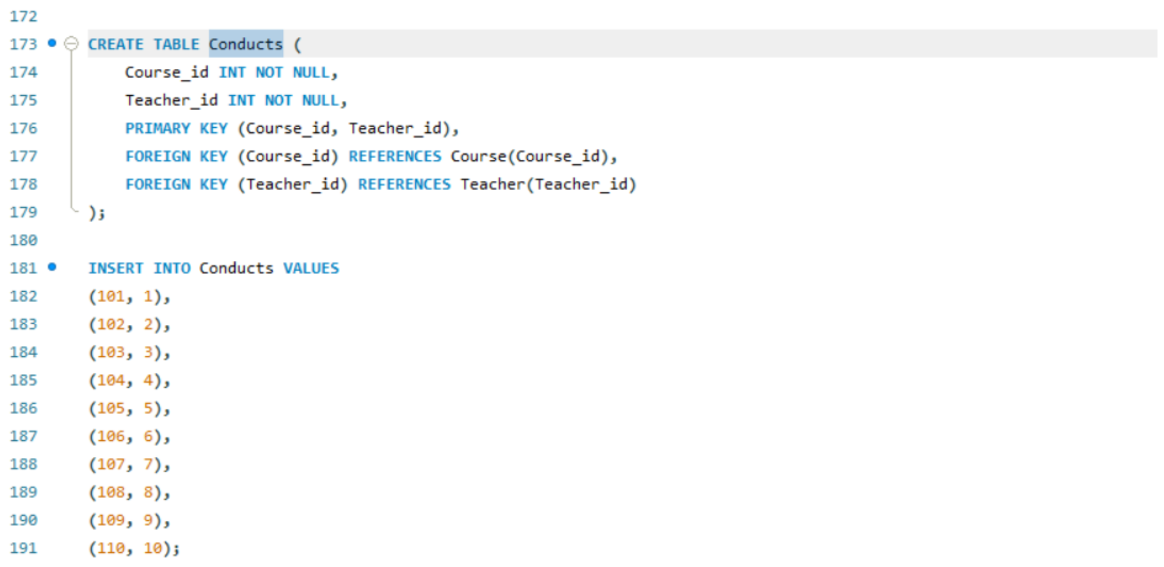
```sql
CREATE TABLE Enrolls (
    Student_id INT NOT NULL,
    Course_id INT NOT NULL,
    PRIMARY KEY (Student_id, Course_id),
    FOREIGN KEY (Student_id) REFERENCES Student(Student_id),
    FOREIGN KEY (Course_id) REFERENCES Course(Course_id)
);

INSERT INTO Enrolls VALUES
(1, 101),
(2, 102),
(3, 103),
(4, 104),
(5, 105),
(6, 106),
(7, 107),
(8, 108),
(9, 109),
(10, 110);
```

**<u>Enrolls Table (Output)</u>:**



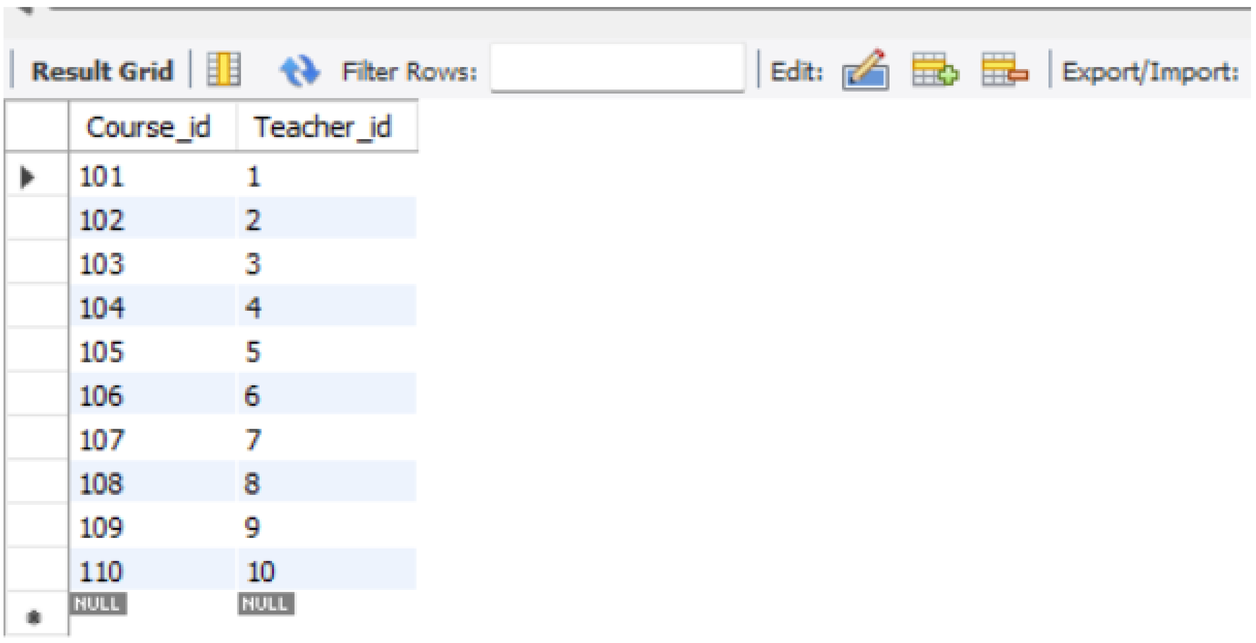
| Student_id | Course_id |
|---|---|
| 1 | 101 |
| 2 | 102 |
| 3 | 103 |
| 4 | 104 |
| 5 | 105 |
| 6 | 106 |
| 7 | 107 |
| 8 | 108 |
| 9 | 109 |
| 10 | 110 |
| NULL | NULL |

## 10. Conducts Table:



```
CREATE TABLE Conducts (
    Course_id INT NOT NULL,
    Teacher_id INT NOT NULL,
    PRIMARY KEY (Course_id, Teacher_id),
    FOREIGN KEY (Course_id) REFERENCES Course(Course_id),
    FOREIGN KEY (Teacher_id) REFERENCES Teacher(Teacher_id)
);

INSERT INTO Conducts VALUES
(101, 1),
(102, 2),
(103, 3),
(104, 4),
(105, 5),
(106, 6),
(107, 7),
(108, 8),
(109, 9),
(110, 10);
```

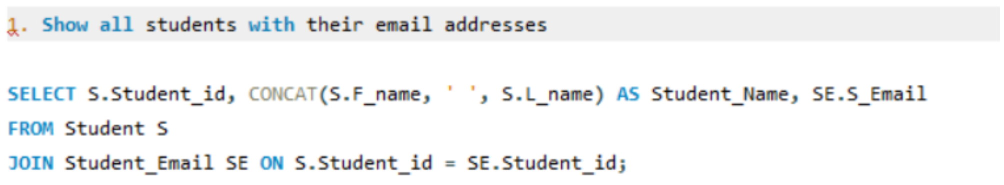
## Conducts Table (Output):



| Course_id | Teacher_id |
|---|---|
| 101 | 1 |
| 102 | 2 |
| 103 | 3 |
| 104 | 4 |
| 105 | 5 |
| 106 | 6 |
| 107 | 7 |
| 108 | 8 |
| 109 | 9 |
| 110 | 10 |
| NULL | NULL |

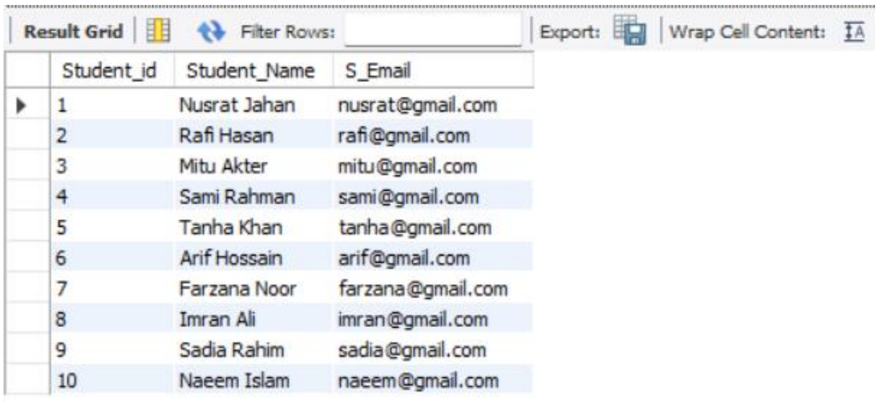
## Logical Sql Queries

**1.**Show all students with their email addresses:



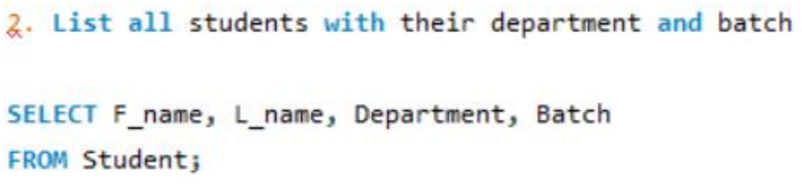
```
1. Show all students with their email addresses

SELECT S.Student_id, CONCAT(S.F_name, ' ', S.L_name) AS Student_Name, SE.S_Email
FROM Student S
JOIN Student_Email SE ON S.Student_id = SE.Student_id;
```

Output:



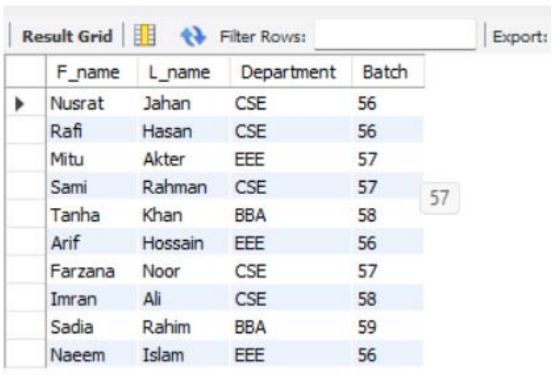
| Student_id | Student_Name | S_Email |
|---|---|---|
| 1 | Nusrat Jahan | nusrat@gmail.com |
| 2 | Rafi Hasan | rafi@gmail.com |
| 3 | Mitu Akter | mitu@gmail.com |
| 4 | Sami Rahman | sami@gmail.com |
| 5 | Tanha Khan | tanha@gmail.com |
| 6 | Arif Hossain | arif@gmail.com |
| 7 | Farzana Noor | farzana@gmail.com |
| 8 | Imran Ali | imran@gmail.com |
| 9 | Sadia Rahim | sadia@gmail.com |
| 10 | Naeem Islam | naeem@gmail.com |

**2.**List all students with their department and batch:



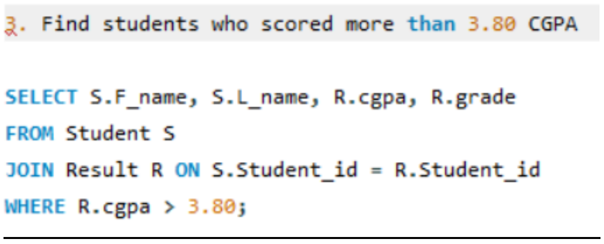
```
2. List all students with their department and batch

SELECT F_name, L_name, Department, Batch
FROM Student;
```

Output:



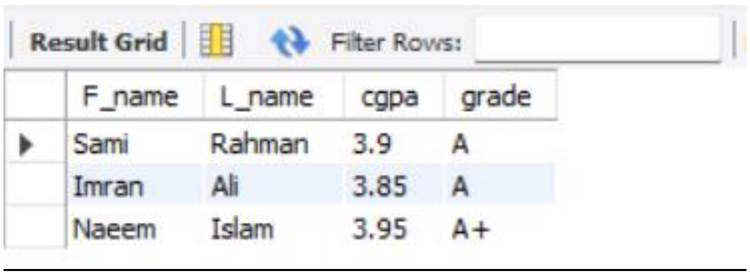
| F_name | L_name | Department | Batch |
|---|---|---|---|
| Nusrat | Jahan | CSE | 56 |
| Rafi | Hasan | CSE | 56 |
| Mitu | Akter | EEE | 57 |
| Sami | Rahman | CSE | 57 |
| Tanha | Khan | BBA | 58 |
| Arif | Hossain | EEE | 56 |
| Farzana | Noor | CSE | 57 |
| Imran | Ali | CSE | 58 |
| Sadia | Rahim | BBA | 59 |
| Naeem | Islam | EEE | 56 |

**3.**Find students who scored more than 3.80 CGPA:



```
3. Find students who scored more than 3.80 CGPA

SELECT S.F_name, S.L_name, R.cgpa, R.grade
FROM Student S
JOIN Result R ON S.Student_id = R.Student_id
WHERE R.cgpa > 3.80;
```

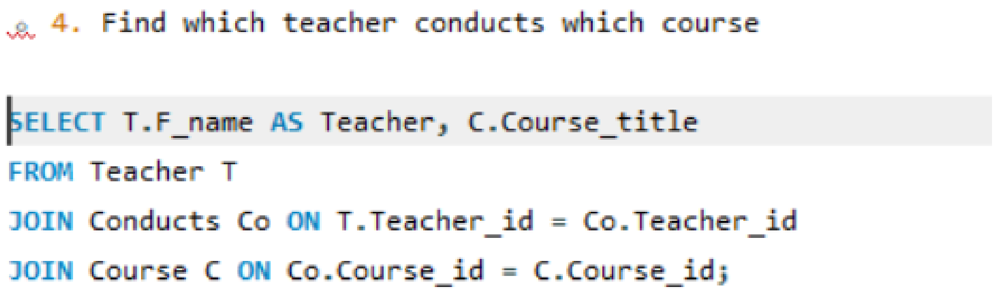
Output:



Result Grid | Filter Rows:

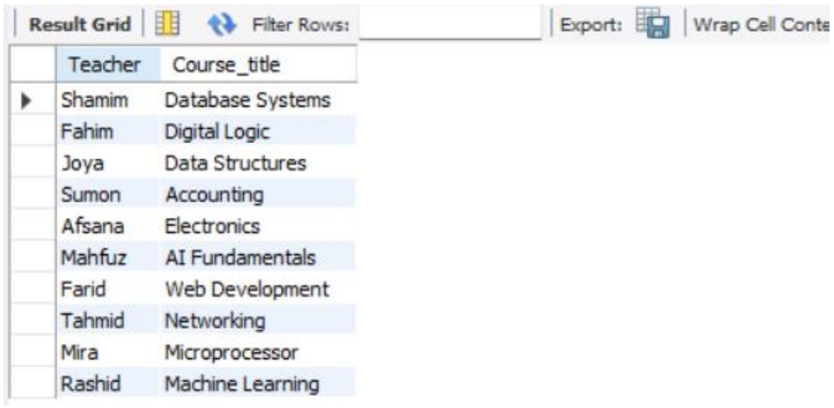
| F_name | L_name | cgpa | grade |
|---|---|---|---|
| Sami | Rahman | 3.9 | A |
| Imran | Ali | 3.85 | A |
| Naeem | Islam | 3.95 | A+ |

**4.**Find which teacher conducts which course:



```
4. Find which teacher conducts which course

SELECT T.F_name AS Teacher, C.Course_title
FROM Teacher T
JOIN Conducts Co ON T.Teacher_id = Co.Teacher_id
JOIN Course C ON Co.Course_id = C.Course_id;
```

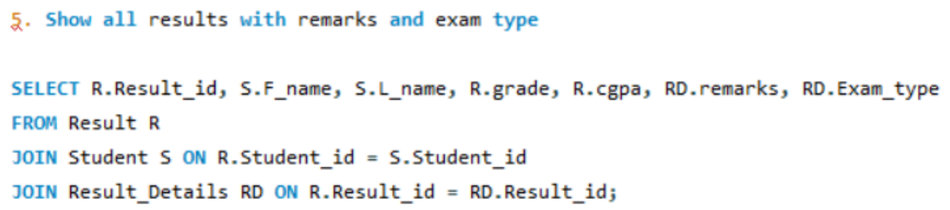
Output:



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

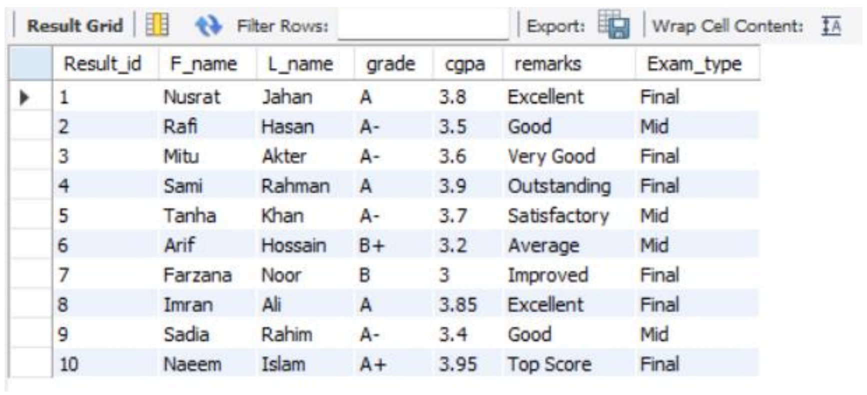
| Teacher | Course_title |
|---|---|
| Shamim | Database Systems |
| Fahim | Digital Logic |
| Joya | Data Structures |
| Sumon | Accounting |
| Afsana | Electronics |
| Mahfuz | AI Fundamentals |
| Farid | Web Development |
| Tahmid | Networking |
| Mira | Microprocessor |
| Rashid | Machine Learning |

**5.**Show all results with remarks and exam type:



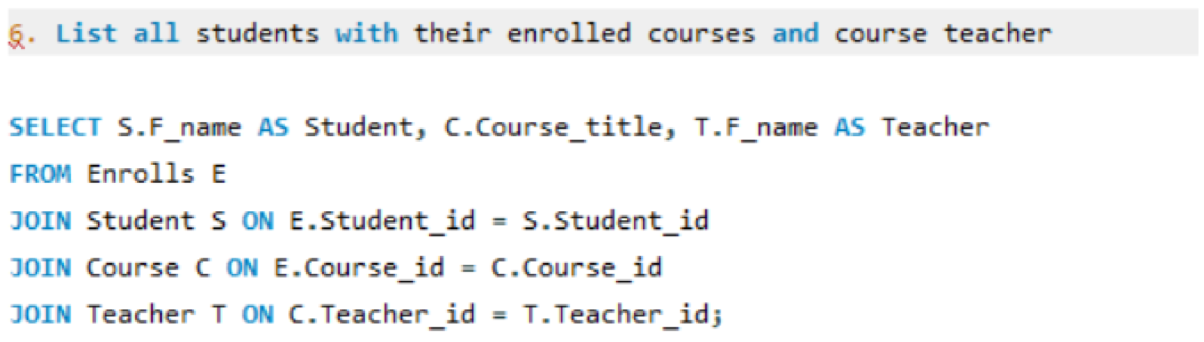
```
5. Show all results with remarks and exam type

SELECT R.Result_id, S.F_name, S.L_name, R.grade, R.cgpa, RD.remarks, RD.Exam_type
FROM Result R
JOIN Student S ON R.Student_id = S.Student_id
JOIN Result_Details RD ON R.Result_id = RD.Result_id;
```

Output:



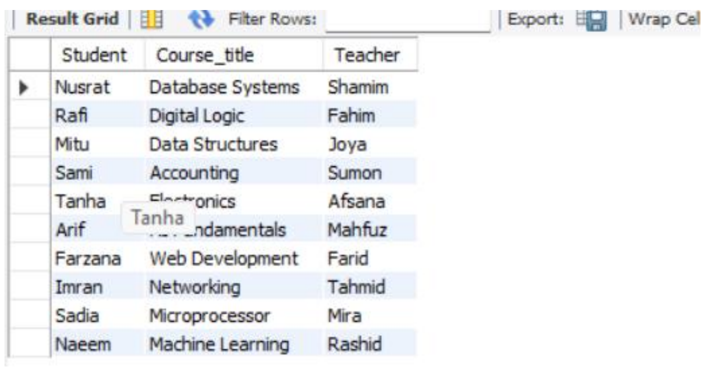
| Result_id | F_name | L_name | grade | cgpa | remarks | Exam_type |
|---|---|---|---|---|---|---|
| 1 | Nusrat | Jahan | A | 3.8 | Excellent | Final |
| 2 | Rafi | Hasan | A- | 3.5 | Good | Mid |
| 3 | Mitu | Akter | A- | 3.6 | Very Good | Final |
| 4 | Sami | Rahman | A | 3.9 | Outstanding | Final |
| 5 | Tanha | Khan | A- | 3.7 | Satisfactory | Mid |
| 6 | Arif | Hossain | B+ | 3.2 | Average | Mid |
| 7 | Farzana | Noor | B | 3 | Improved | Final |
| 8 | Imran | Ali | A | 3.85 | Excellent | Final |
| 9 | Sadia | Rahim | A- | 3.4 | Good | Mid |
| 10 | Naeem | Islam | A+ | 3.95 | Top Score | Final |

**6.**List all students with their enrolled courses and course teacher:



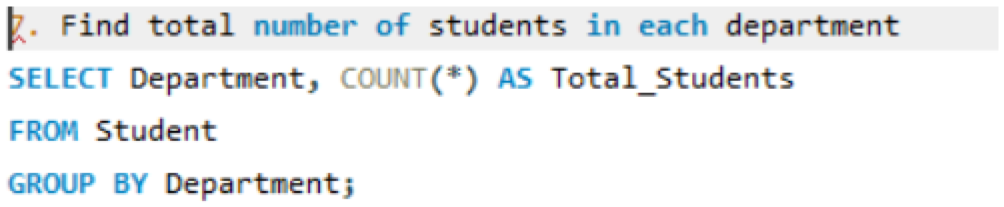
```
6. List all students with their enrolled courses and course teacher

SELECT S.F_name AS Student, C.Course_title, T.F_name AS Teacher
FROM Enrolls E
JOIN Student S ON E.Student_id = S.Student_id
JOIN Course C ON E.Course_id = C.Course_id
JOIN Teacher T ON C.Teacher_id = T.Teacher_id;
```

Output:



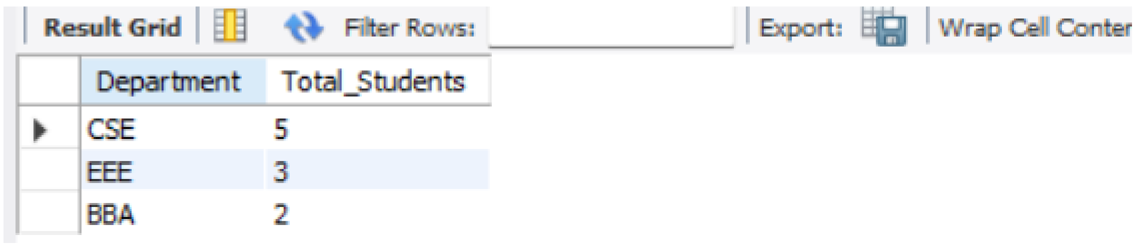
| Student | Course_title | Teacher |
|---|---|---|
| Nusrat | Database Systems | Shamim |
| Rafi | Digital Logic | Fahim |
| Mitu | Data Structures | Joya |
| Sami | Accounting | Sumon |
| Tanha | Electronics | Afsana |
| Arif | [illegible]ndamentals | Mahfuz |
| Farzana | Web Development | Farid |
| Imran | Networking | Tahmid |
| Sadia | Microprocessor | Mira |
| Naeem | Machine Learning | Rashid |

**7.**Find total number of students in each department:



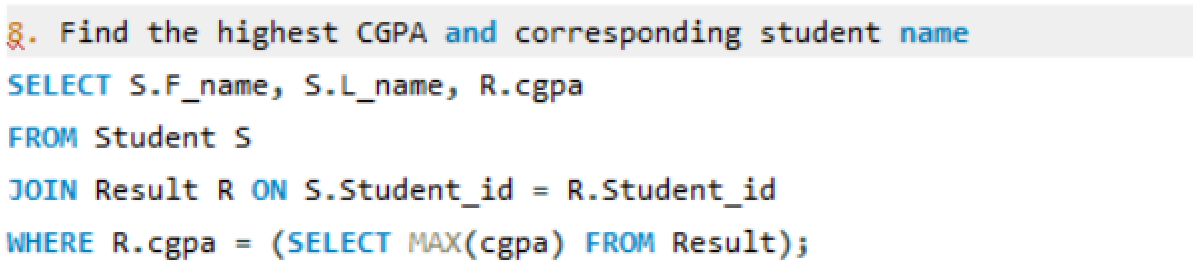
```
7. Find total number of students in each department
SELECT Department, COUNT(*) AS Total_Students
FROM Student
GROUP BY Department;
```

Output:



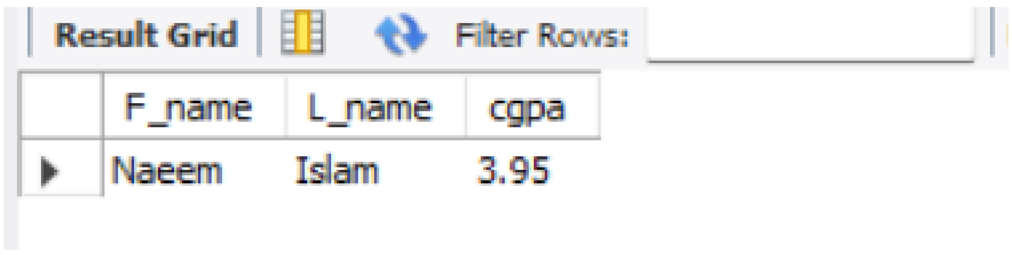
| Department | Total_Students |
|---|---|
| CSE | 5 |
| EEE | 3 |
| BBA | 2 |

**8.**Find the highest CGPA and corresponding student name:



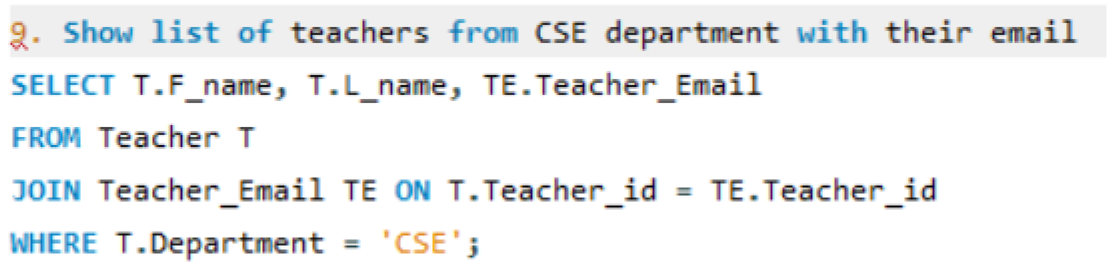
```
8. Find the highest CGPA and corresponding student name
SELECT S.F_name, S.L_name, R.cgpa
FROM Student S
JOIN Result R ON S.Student_id = R.Student_id
WHERE R.cgpa = (SELECT MAX(cgpa) FROM Result);
```

Output:



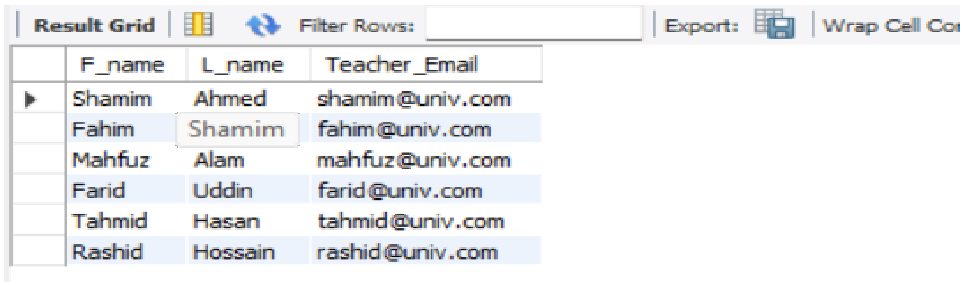
| F_name | L_name | cgpa |
|---|---|---|
| Naeem | Islam | 3.95 |

**9.**Show list of teachers from CSE department with their email:



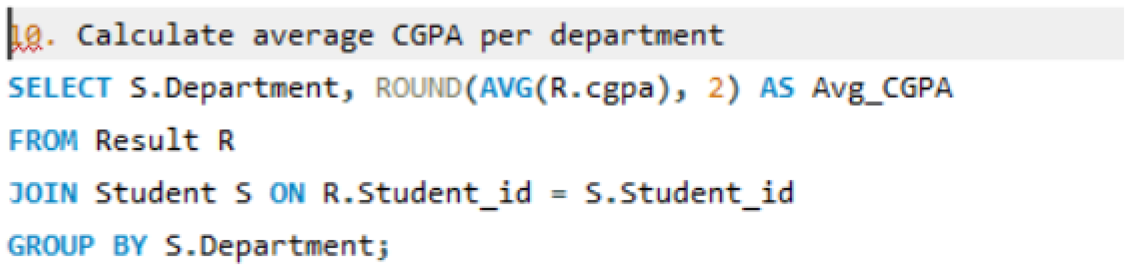
```
9. Show list of teachers from CSE department with their email
SELECT T.F_name, T.L_name, TE.Teacher_Email
FROM Teacher T
JOIN Teacher_Email TE ON T.Teacher_id = TE.Teacher_id
WHERE T.Department = 'CSE';
```

Output:



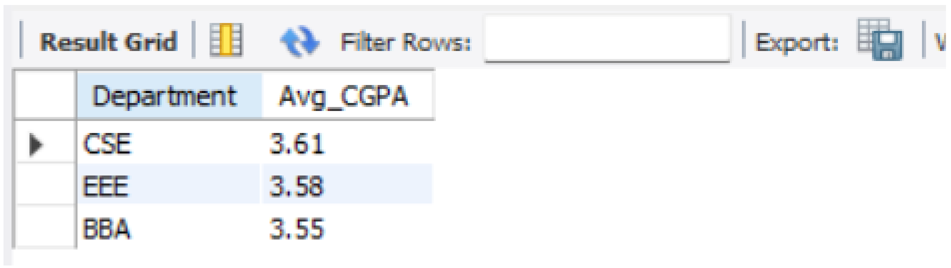
| F_name | L_name | Teacher_Email |
|---|---|---|
| Shamim | Ahmed | shamim@univ.com |
| Fahim | Shamim | fahim@univ.com |
| Mahfuz | Alam | mahfuz@univ.com |
| Farid | Uddin | farid@univ.com |
| Tahmid | Hasan | tahmid@univ.com |
| Rashid | Hossain | rashid@univ.com |

**10.**Calculate average CGPA per department:



```
10. Calculate average CGPA per department
SELECT S.Department, ROUND(AVG(R.cgpa), 2) AS Avg_CGPA
FROM Result R
JOIN Student S ON R.Student_id = S.Student_id
GROUP BY S.Department;
```

Output:



| Department | Avg_CGPA |
|---|---|
| CSE | 3.61 |
| EEE | 3.58 |
| BBA | 3.55 |

## Outcome

- ❖ Error minimization and faster result processing.
- ❖ Improved data security and accuracy.
- ❖ Enhanced user accessibility via different result viewing formats.
- ❖ Reduced administrative workload and paper consumption.

## Discussion

The Student Result Management System helps automate result processing and reduce human errors. It ensures accurate data handling and easy access for both teachers and students. By using a well-structured database with proper relationships, the system maintains data consistency and improves overall efficiency in managing academic results.

## References

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2020). *Operating System Concepts* (10th ed.). Wiley.
2. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson.
3. Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems* (3rd ed.). McGraw-Hill.
4. W3Schools. (n.d.). *SQL Tutorial*. Retrieved from https://www.w3schools.com/sql/
5. TutorialsPoint. (n.d.). *DBMS Tutorial*. Retrieved from https://www.tutorialspoint.com/dbms/
6. GeeksforGeeks. (n.d.). *Student Result Management System Project*. Retrieved from https://www.geeksforgeeks.org/
7. GitHub. (n.d.). *Student Result Management System Projects*. Retrieved from https://github.com/
8. IEEE Xplore Digital Library. (n.d.). *Research Papers on Student Result Management Systems*. Retrieved from https://ieeexplore.ieee.org/

.

## Git Hub Links

https://github.com/tamim65k/Student-Result-Management-System-DBMS/
https://github.com/tamim65k
https://github.com/Montasirpeal
https://github.com/paransha1
https://github.com/nusrattasfi
https://github.com/eraboti1031