Report of Assignment

Student Grading Management

I. Description of the database

Many students can enter a group and one group can be managed by many teachers

Each Group\_Students has many Assessment

Each Assessment grades the subject for each student

Each Assessment has different result

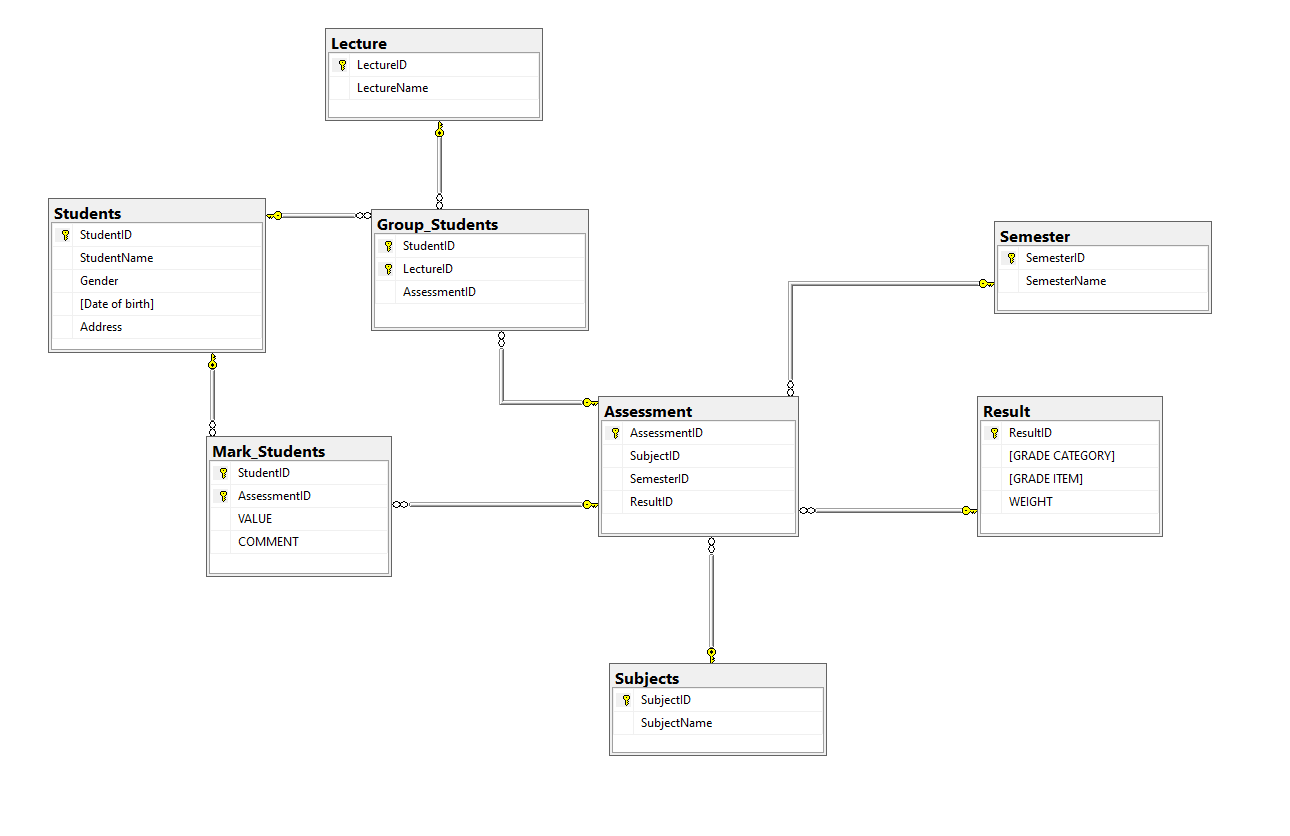
Each Semester has more than one Assessment

Each Subject has more than one Assessment

II. An ERD that fully describes the database.



III. The relational schema derived from the ERD



IV. The input of database statements used to create the tables.

CREATE TABLE Students(

StudentID INT NOT NULL,

StudentName VARCHAR (20) NOT NULL,

Gender BIT NOT NULL,

[Date of birth] DATE NOT NULL,

[Address] VARCHAR(20) NOT NULL,

PRIMARY KEY (StudentID)

);

CREATE TABLE Lecture(

LectureID INT NOT NULL,

LectureName VARCHAR (20) NOT NULL,

PRIMARY KEY (LectureID)

);

CREATE TABLE Subjects(

SubjectID INT NOT NULL,

SubjectName VARCHAR (20) NOT NULL,

PRIMARY KEY (SubjectID),

);

CREATE TABLE Result(

ResultID INT NOT NULL,

[GRADE CATEGORY] VARCHAR(50) NOT NULL,

[GRADE ITEM] VARCHAR(50) NOT NULL,

[WEIGHT] VARCHAR(50) NOT NULL,

PRIMARY KEY (ResultID),

);

CREATE TABLE Semester(

SemesterID INT NOT NULL,

SemesterName VARCHAR (20) NOT NULL,

PRIMARY KEY (SemesterID),

);

CREATE TABLE Assessment(

AssessmentID INT NOT NULL,

SubjectID INT NOT NULL,

SemesterID INT NOT NULL,

ResultID INT NOT NULL,

Category VARCHAR(50) NOT NULL,

[Type] VARCHAR(10) NOT NULL,

Part INT NOT NULL,

[Weight] VARCHAR(10) NOT NULL,

[Completion Criteria] VARCHAR(10) NOT NULL,

Duration VARCHAR(15) NOT NULL,

LO INT,

[Question Type] TEXT NOT NULL,

[No Question] INT,

[Knowledge and Skill] VARCHAR(150) NOT NULL,

[Grading Guide] TEXT NOT NULL,

Note VARCHAR(2000),

PRIMARY KEY (AssessmentID),

FOREIGN KEY (SubjectID) REFERENCES Subjects (SubjectID),

FOREIGN KEY (ResultID) REFERENCES Result (ResultID),

FOREIGN KEY (SemesterID) REFERENCES Semester (SemesterID),

);

CREATE TABLE Group\_Students(

StudentID INT NOT NULL,

LectureID INT NOT NULL,

AssessmentID INT NOT NULL,

PRIMARY KEY (StudentID,LectureID),

FOREIGN KEY (StudentID) REFERENCES Students (StudentID),

FOREIGN KEY (LectureID) REFERENCES Lecture (LectureID),

FOREIGN KEY (AssessmentID) REFERENCES Assessment (AssessmentID)

);

CREATE TABLE Mark\_Students(

StudentID INT NOT NULL,

AssessmentID INT NOT NULL,

[VALUE] FLOAT NOT NULL,

COMMENT VARCHAR(200),

PRIMARY KEY (StudentID),

FOREIGN KEY (StudentID) REFERENCES Students (StudentID),

FOREIGN KEY (AssessmentID) REFERENCES Assessment (AssessmentID),

);

V. 10 queries that demonstrate the usefulness of the database.

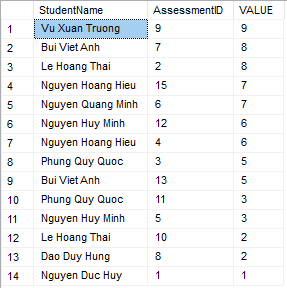
--Mark of each assessment

SELECT s.StudentName, ms.AssessmentID, ms.VALUE

FROM Mark\_Students ms INNER JOIN Students s ON s.StudentID = ms.StudentID

ORDER BY ms.VALUE desc

Result:

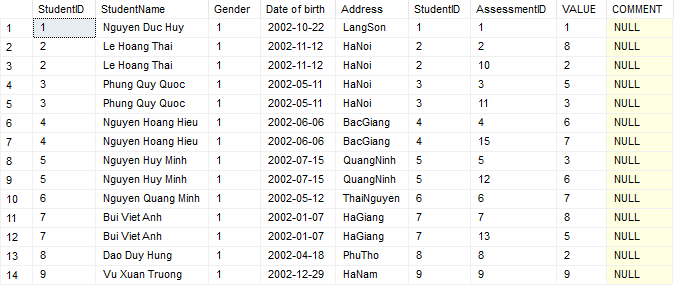


--Infomation and mark of each students

SELECT \*

FROM Students s INNER JOIN Mark\_Students ms ON s.StudentID = ms.StudentID

Result:



--Total assessment of each students

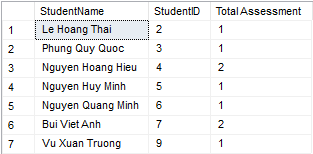
SELECT s.StudentName,ms.StudentID,COUNT(ms.AssessmentID) AS [Total Assessment]

FROM Mark\_Students ms INNER JOIN Students s ON s.StudentID = ms.StudentID

WHERE ms.VALUE >= 5

GROUP BY ms.StudentID,s.StudentName

Result:



--Total assessment student done and total mark of them

SELECT s.StudentName,ms.StudentID,COUNT(ms.AssessmentID) AS [Number of Assessment],SUM(ms.VALUE) AS [Total Point of All Assessment]

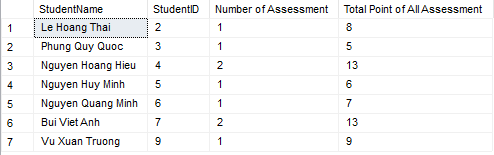
FROM Mark\_Students ms INNER JOIN Students s ON s.StudentID = ms.StudentID

WHERE ms.VALUE >= 5

GROUP BY ms.StudentID,s.StudentName

HAVING SUM(ms.VALUE) > 2

Result:



--Total mark all assessment of each students and mark value > 5

WITH t AS

(

SELECT s.StudentID, s.StudentName, ms.AssessmentID,SUM(ms.VALUE) AS [Value of Assessment]

FROM Students s, Mark\_Students ms

WHERE s.StudentID = ms.StudentID

GROUP BY s.StudentID,s.StudentName,ms.AssessmentID

)

SELECT t.StudentID, t.StudentName, t.[Value of Assessment]

FROM t, Assessment a

WHERE t.AssessmentID = a.AssessmentID AND t.[Value of Assessment] > 5

ORDER BY t.[Value of Assessment] DESC

Result:



--Find subject in semester 3 and have name is ‘LAB211’

SELECT s.SubjectID,s.SubjectName

FROM Assessment a INNER JOIN Result r ON a.ResultID = r.ResultID

INNER JOIN Subjects s ON a.SubjectID = s.SubjectID

WHERE a.SemesterID = 3 AND s.SubjectName = 'LAB211'

ORDER BY s.SubjectID

Result:



--Find group students have supervise is Sonnt

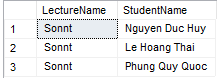
SELECT l.LectureName, s.StudentName

FROM Lecture l INNER JOIN Group\_Students gs ON l.LectureID = gs.LectureID

INNER JOIN Students s ON gs.StudentID = s.StudentID

WHERE l.LectureName = 'Sonnt'

Result:



--Find infomation of assessment in semester 3 and have id result is 3

tim ra thong tin cua assessment nam trong ki 3 va co id ket qua la 3

SELECT \*

FROM Assessment a1 INNER JOIN Assessment a2 ON a1.AssessmentID = a2.AssessmentID

WHERE a2.SemesterID = 3 AND a1.ResultID = 3

Result:



--TRIGGER : Delete mark of student and replace new mark

CREATE TRIGGER delete\_mark ON Mark\_Students

AFTER DELETE

AS

BEGIN

INSERT INTO Mark\_Students (StudentID,AssessmentID,VALUE)

VALUES (1,1,1)

END

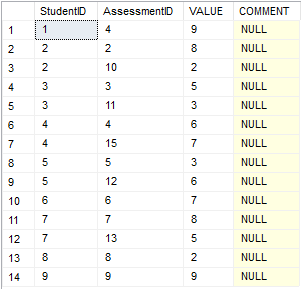
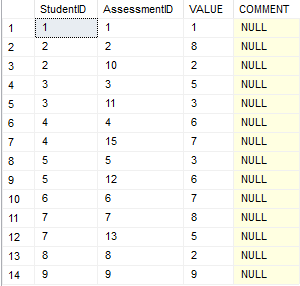
DELETE FROM Mark\_Students

WHERE StudentID = 1

SELECT \* FROM Mark\_Students

Result:

Before After

--STORE PROCEDURE: Information and supervise of each students

CREATE PROCEDURE info\_student AS

SELECT s.StudentID,s.StudentName,s.Gender,s.[Date of birth],s.Address,l.LectureName

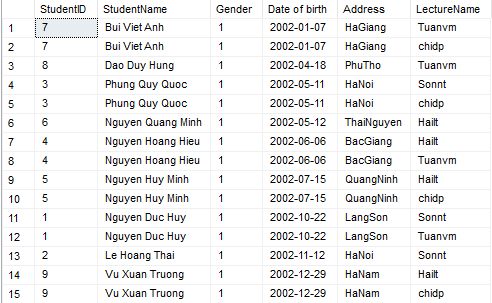
FROM Students s INNER JOIN Group\_Students gs ON s.StudentID = gs.StudentID

INNER JOIN Lecture l ON gs.LectureID = l.LectureID

ORDER BY s.[Date of birth]

EXEC info\_student

Result:



--INDEX: Display infomation of mark students table

CREATE INDEX find\_min\_mark\_student ON [Mark\_Students](StudentID)

SELECT \* FROM Mark\_Students

Result:

