Assignment – Attendance Taking & Student Grade Management System at FPT University Database System

HE160625-AI1604-Nguyen Hai Khang, Lecturer Ngo Tung Son

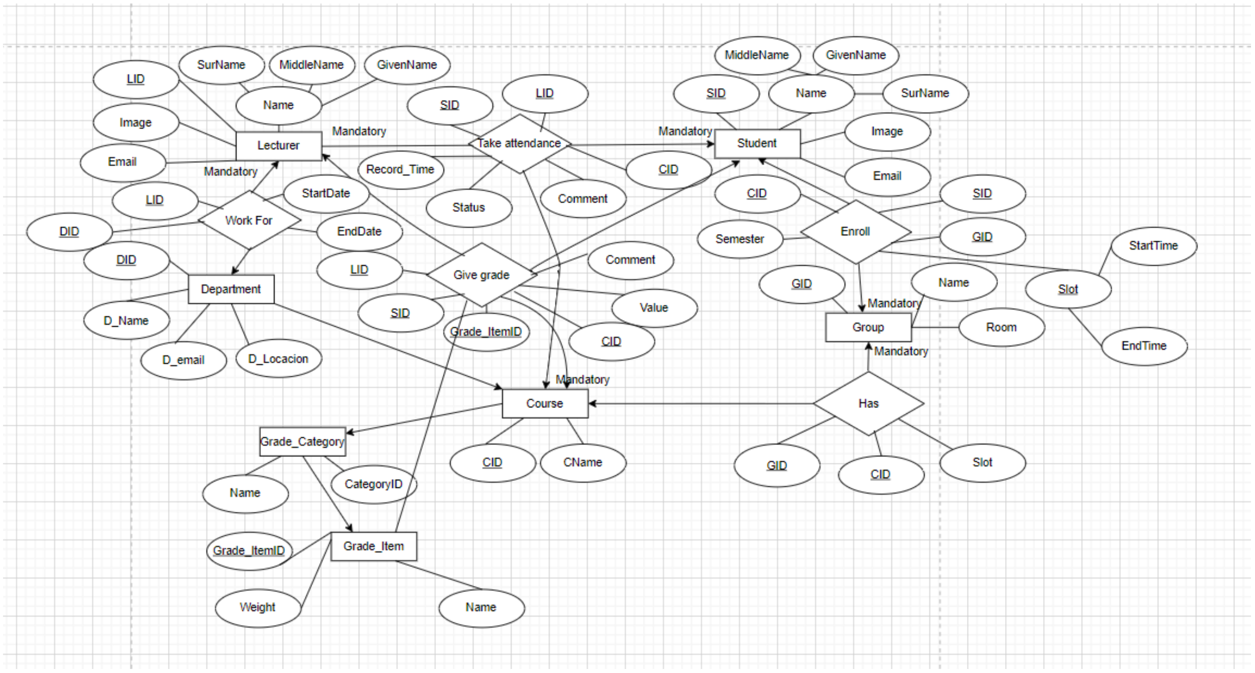
In FPT university, all student use fap.fpt.edu.vn website to do all their works about education. Attendance taking and Grade management system is two of many things in FAP. Another things are: school fee management, student request management… I think there are some application for Lecturer in FAP.

In my business between me and Mr. Sonnt, I will create attendance taking & student grade management system at FPT University DBMS.

A. Descript of database

To create the DBMS for attendance taking & student grade management system at FPT University, the Object must be defined first. The core object is Lecturer, Student and Course. But it not enough, there are many things will concern: departments, grade items, group(class). And many relationships between them. I have created 14 tables (requirement is at least 6) but is not all about attendance taking & student grade management system at FPT University. The next step I did is create ERD (I will explain in B). Create relational schema from ERD in 3NF. Create DBMS in MSQL, and wrote 10 queries and trigger, procedure.

B. ERD **(Entity Relationship Diagram)**

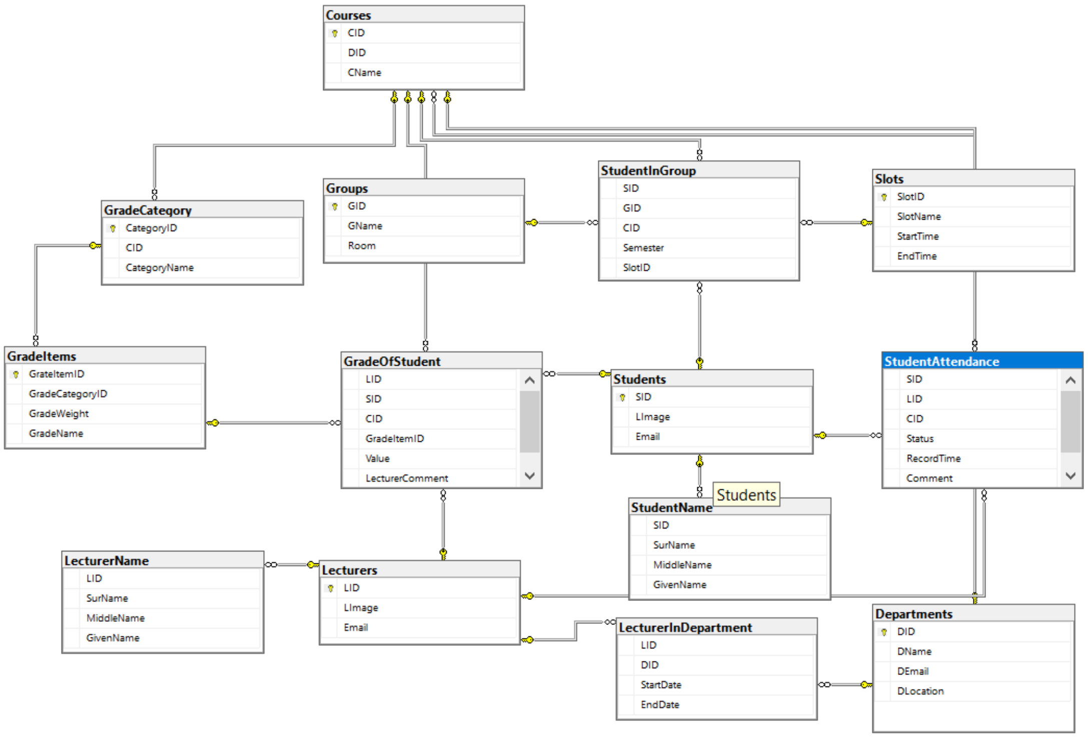


There are 3 main objects in here: Lecturer, Student Course and Grade. Department managers Lecturers, Students, Courses by puts everything in groups.

I will explain the relationship between them:

1. The First relationship in this DB: Lecturer and Department, sometime it is one to many: a department have many Lecturers. But in this case, I think it's is many to many relationships. I have created many to many relationship. Because, I think my lecturer (Mr. SonNT) can work for more than one department each semester. In this relationship, Lecturer is required, because there are no Lecturer in FU who not work for a department.
2. **The main relationship of this project**: (Attendance management and grade management for Students). This is many to many relationship: In "give grade" relationship: Lecturer, Student, CourseID, Course grade type, Value are mandatory. Because: a grade can't be give if it has No lecturer or student or Course, or grade type, or value
3. The relationship between Students and Group: In each semester, Students enroll in a group, this relationship is many to many relationship. Because each group has many students and each student can enroll to many groups. In this relationship, group is mandatory because when enough student department will create group.
4. **The main relationship of this project**: Attendance. In each Course's slot, Lecturer will check attendance for student. This is many to many relationship because: a Lecturer can check attendance for many students, and a students can be checked by many Lecturers.In this relationship: Student, Lecturer, Course, Status (attended or absent), RecordTime are required.But RecordTime is auto saved in some programing frame work like PHP Laravel.In UI, it is a simple way to do this, Lectured open a list of student. Check status and click OK. That will be done.

C. The relation schema derived from the ERD in 3NF:



Step1: find every entity and create corresponding table: The entity that I found are: Lecturer, Student, Departments, Courses, Course grade item.

Step2: Find all one - to many relationship. For each, add FK column to many side table that refer to PK column in one side table. One to many relationship: a course have many course grade item, a course have many slots.

Step3: Find all many to many relationship. For each create new table contain FK to both side table: many to many relationship:

- Create the table LecturerInDepartment for relationship “work for” between Lecturer and Department.

- Create table StudentInGroup for many to many relationship “Enroll” between Students and Group.

- Create table StudentAttendance for many to many relationship “Take attendance” between Lecturer, Student and Course.

- Create table GradeOfStudent for many to many relationship “give grade” between Lecturer, Student and Course.

Step4: Find all weak entity, create it PK as combination of it key attribute and dependence FK attribute: There are no weak entity in my ERD

Step 5: Find one to one Relationship use strategy: Do exactly like step 3 or add the FK to un mandatory table: in this table one – one relation ship is Student and StudentsName, I use the strategy add the FK to un mandatory table. Because the student's name is not change or it is very rare.

Step 6: Fill all multiple part attribute, foreach split them into several columns corresponding to its parts: Separate Student name and Lecturer name in StudentName and LecturedName table

Step7: Fill all multiple value attribute, foreach apply similar strategy to weak entity: there are no multiple value in my ERD

D. The set of database statement used to create the tables used in my database.

I will put them in gifthub. The below are some example:

Create and Use DB:

```
CREATE DATABASE Asignment2_AI1604_KhangNH_Lecturer_SonNT
USE Asignment2_AI1604_KhangNH_Lecturer_SonNT
```

Create table:

```
CREATE TABLE Lecturers(
       LID INT IDENTITY(1,1) PRIMARY KEY,
       LImage VARBINARY(MAX),
       Email VARCHAR(255)
);
```

Insert value into table

```
INSERT INTO Lecturers(LImage,Email) VALUES (00112111 ,'Sonnt@fpt.edu.vn')
INSERT INTO Lecturers(LImage,Email) VALUES (0011121233411 ,'Sonnt5@fpt.edu.vn')
INSERT INTO Lecturers(LImage,Email) VALUES (0011121233411 ,'TuNT57@fpt.edu.vn')
INSERT INTO Lecturers(LImage,Email) VALUES (0011211123 ,'phake1@fpt.edu.vn')
```

Update table

```
UPDATE Lecturers
SET Email = 'phake123@gmail.com'
WHERE LID = 1
```

DELETE record in table:

```
DELETE FROM Lecturers WHERE LID = 1
```

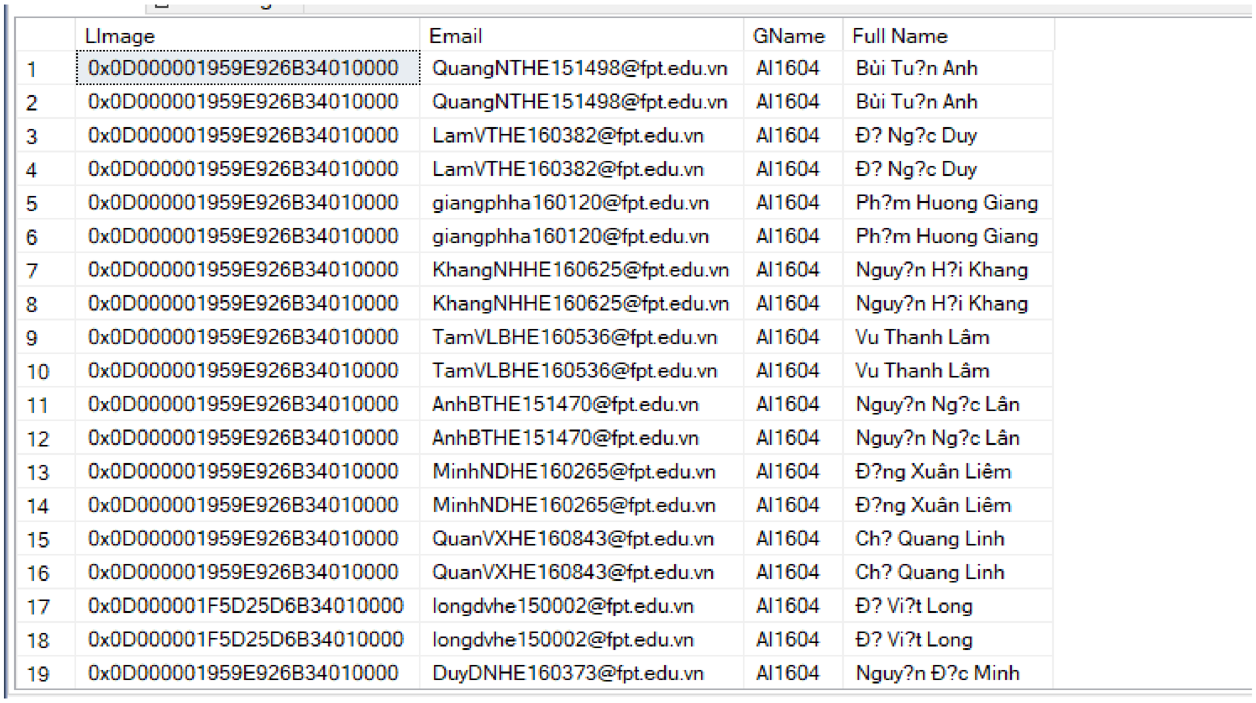
E: 10 queries that demonstrate the usefulness of the database

1.

```
-- Query 1: A query that uses ORDER BY -- display name of students in a class ordered by
name ASC
SELECT s.LImage, s.Email, g.GName, sn.SurName + ' ' + sn.MiddleName + ' ' + sn.GivenName
AS [Full Name] FROM Students s
INNER JOIN StudentInGroup sg ON s.SID = sg.SID
INNER JOIN Groups g ON sg.GID = g.GID
INNER JOIN StudentName sn ON sn.SID = s.SID
```

```sql
ORDER BY sn.GivenName ASC
```

Result:

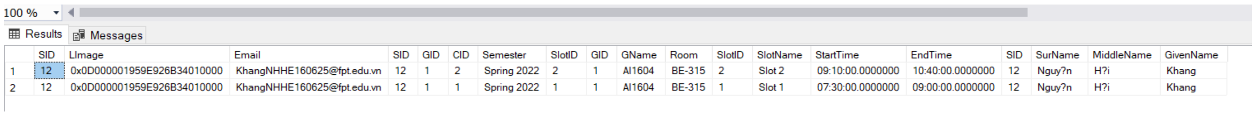


	LImage	Email	GName	Full Name
1	0x0D000001959E926B34010000	QuangNTHE151498@fpt.edu.vn	AI1604	Bùi Tu?n Anh
2	0x0D000001959E926B34010000	QuangNTHE151498@fpt.edu.vn	AI1604	Bùi Tu?n Anh
3	0x0D000001959E926B34010000	LamVTHE160382@fpt.edu.vn	AI1604	Đ? Ng?c Duy
4	0x0D000001959E926B34010000	LamVTHE160382@fpt.edu.vn	AI1604	Đ? Ng?c Duy
5	0x0D000001959E926B34010000	giangphha160120@fpt.edu.vn	AI1604	Ph?m Huong Giang
6	0x0D000001959E926B34010000	giangphha160120@fpt.edu.vn	AI1604	Ph?m Huong Giang
7	0x0D000001959E926B34010000	KhangNHHE160625@fpt.edu.vn	AI1604	Nguy?n H?i Khang
8	0x0D000001959E926B34010000	KhangNHHE160625@fpt.edu.vn	AI1604	Nguy?n H?i Khang
9	0x0D000001959E926B34010000	TamVLBHE160536@fpt.edu.vn	AI1604	Vu Thanh Lâm
10	0x0D000001959E926B34010000	TamVLBHE160536@fpt.edu.vn	AI1604	Vu Thanh Lâm
11	0x0D000001959E926B34010000	AnhBTHE151470@fpt.edu.vn	AI1604	Nguy?n Ng?c Lân
12	0x0D000001959E926B34010000	AnhBTHE151470@fpt.edu.vn	AI1604	Nguy?n Ng?c Lân
13	0x0D000001959E926B34010000	MinhNDHE160265@fpt.edu.vn	AI1604	Đ?ng Xuân Liêm
14	0x0D000001959E926B34010000	MinhNDHE160265@fpt.edu.vn	AI1604	Đ?ng Xuân Liêm
15	0x0D000001959E926B34010000	QuanVXHE160843@fpt.edu.vn	AI1604	Ch? Quang Linh
16	0x0D000001959E926B34010000	QuanVXHE160843@fpt.edu.vn	AI1604	Ch? Quang Linh
17	0x0D000001F5D25D6B34010000	longdvhe150002@fpt.edu.vn	AI1604	Đ? Vi?t Long
18	0x0D000001F5D25D6B34010000	longdvhe150002@fpt.edu.vn	AI1604	Đ? Vi?t Long
19	0x0D000001959E926B34010000	DuyDNHE160373@fpt.edu.vn	AI1604	Nguy?n Đ?c Minh

2.

```sql
-- Query 2:A query that uses INNER JOINS - A schedule of a student name is Khang. It
matches with view schedule in FAP
-- This query will return a object array, Front End programer will use it and display
like user interface
-- Some thing went wrong here, because there no slot detail for each course. I don't have
the table like this, but it will be OK. There is a sample
-- To show the detail like FAP, I must create a table like course - slot. a course have
30 slot, and each slot having a Slot ID, and a day of each slot.
SELECT * FROM Students s
INNER JOIN StudentInGroup sg ON s.SID = sg.SID
INNER JOIN Groups g ON sg.GID = g.GID
INNER JOIN Slots sl ON sg.SlotID = sl.SlotID
INNER JOIN StudentName sn ON sn.SID = s.SID
WHERE sn.GivenName = 'Khang'
```

Result:

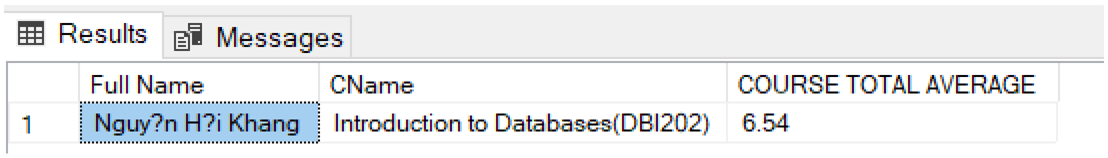


	SID	LImage	Email	SID	GID	CID	Semester	SlotID	GID	GName	Room	SlotID	SlotName	StartTime	EndTime	SID	SurName	MiddleName	GivenName
1	12	0x0D000001959E926B34010000	KhangNHHE160625@fpt.edu.vn	12	1	2	Spring 2022	2	1	AI1604	BE-315	2	Slot 2	09:10:00.0000000	10:40:00.0000000	12	Nguy?n	H?i	Khang
2	12	0x0D000001959E926B34010000	KhangNHHE160625@fpt.edu.vn	12	1	1	Spring 2022	1	1	AI1604	BE-315	1	Slot 1	07:30:00.0000000	09:00:00.0000000	12	Nguy?n	H?i	Khang

3.

```sql
--  A query that uses aggregate functions: Caculate average point of a student in a course
SELECT sn.SurName + ' ' + sn.MiddleName + ' ' + sn.GivenName AS [Full Name], c.CName,
SUM(gs.Value * gi.GradeWeight) AS [COURSE TOTAL AVERAGE] FROM Students s
INNER JOIN GradeOfStudent gs ON s.SID = gs.SID
INNER JOIN Courses c ON C.CID = gs.CID
INNER JOIN GradeItems gi ON gi.GrateItemID = gs.GradeItemID
INNER JOIN StudentName sn ON s.SID = sn.SID
WHERE sn.GivenName = 'Khang'
GROUP BY sn.SurName + ' ' + sn.MiddleName + ' ' + sn.GivenName, c.CName
```

Result:



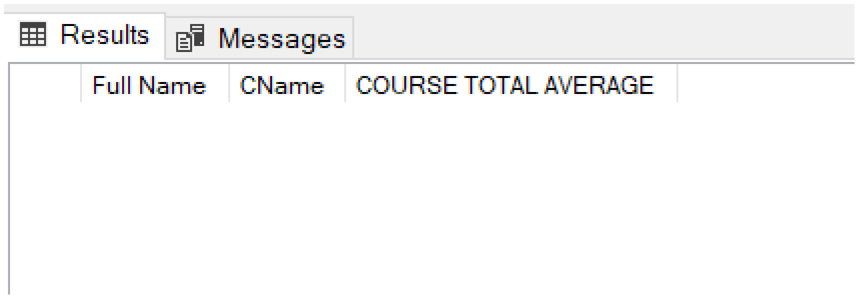
Results | Messages

	Full Name	CName	COURSE TOTAL AVERAGE
1	Nguy?n H?i Khang	Introduction to Databases(DBI202)	6.54

4.

```sql
--  A query that uses the GROUP BY and HAVING clauses: Show the good student in DBI202 who having course average  >= 8.0.
-- This query return a empty record, because there are no student having GPA >= 8
SELECT sn.SurName + ' ' + sn.MiddleName + ' ' + sn.GivenName AS [Full Name], c.CName,
SUM(gs.Value * gi.GradeWeight) AS [COURSE TOTAL AVERAGE] FROM Students s
INNER JOIN GradeOfStudent gs ON s.SID = gs.SID
INNER JOIN Courses c ON C.CID = gs.CID
INNER JOIN GradeItems gi ON gi.GrateItemID = gs.GradeItemID
INNER JOIN StudentName sn ON s.SID = sn.SID
GROUP BY sn.SurName + ' ' + sn.MiddleName + ' ' + sn.GivenName, c.CName
HAVING SUM(gs.Value * gi.GradeWeight) >= 8.0
```

Result:



Results | Messages

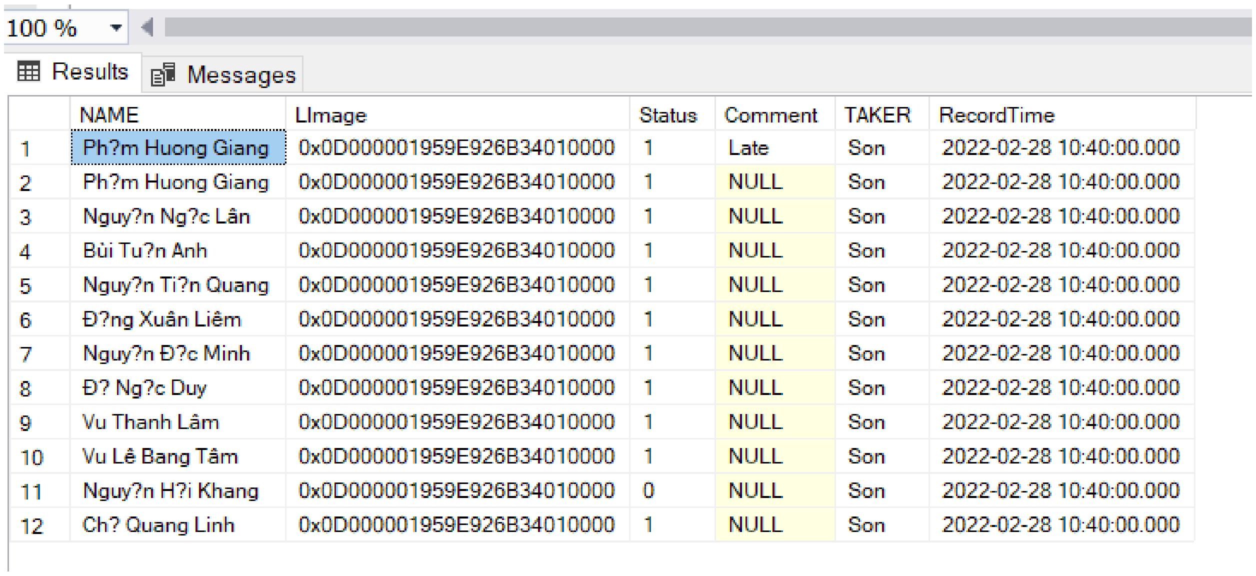
	Full Name	CName	COURSE TOTAL AVERAGE

5.

```
-- A query that uses a sub-query as a relation: show single activity attendance:

SELECT StudentNametbl.NAME, StudentNametbl.LImage, sa.Status, sa.Comment,
LecturerNametbl.TAKER, sa.RecordTime FROM StudentAttendance sa INNER JOIN
(SELECT s.SID AS SID, sn.SurName + ' ' + sn.MiddleName + ' ' + sn.GivenName AS [NAME],
s.LImage FROM Students s INNER JOIN StudentName sn on s.SID = sn.SID)StudentNametbl
ON sa.SID = StudentNametbl.SID
INNER JOIN
(SELECT l.LID as LID, ln.GivenName as TAKER FROM Lecturers l INNER JOIN LecturerName ln
ON l.LID = ln.LID)LecturerNametbl
ON sa.LID = LecturerNametbl.LID
```

Result:



100 %

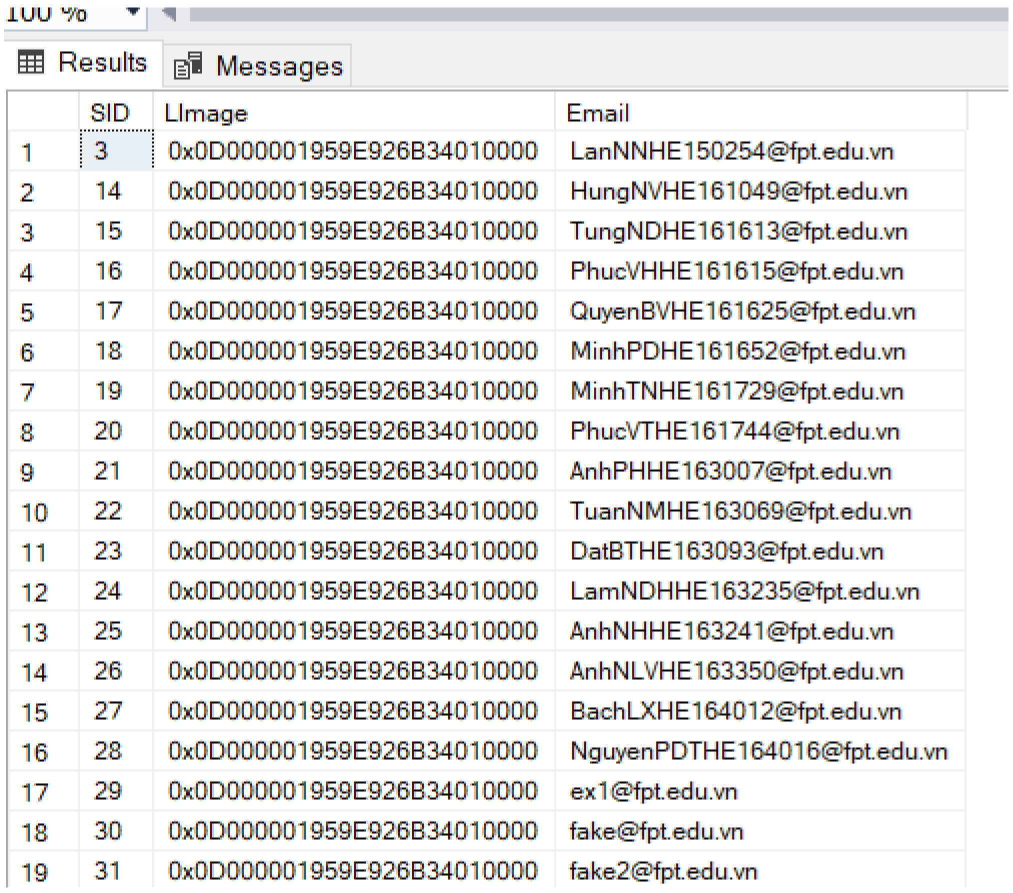
Results | Messages

	NAME	LImage	Status	Comment	TAKER	RecordTime
1	Ph?m Huong Giang	0x0D000001959E926B34010000	1	Late	Son	2022-02-28 10:40:00.000
2	Ph?m Huong Giang	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000
3	Nguy?n Ng?c Lân	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000
4	Bùi Tu?n Anh	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000
5	Nguy?n Ti?n Quang	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000
6	Đ?ng Xuân Liêm	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000
7	Nguy?n Đ?c Minh	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000
8	Đ? Ng?c Duy	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000
9	Vu Thanh Lâm	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000
10	Vu Lê Bang Tâm	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000
11	Nguy?n H?i Khang	0x0D000001959E926B34010000	0	NULL	Son	2022-02-28 10:40:00.000
12	Ch? Quang Linh	0x0D000001959E926B34010000	1	NULL	Son	2022-02-28 10:40:00.000

6.

```
--  A query that uses a sub-query in the WHERE clause: show the student who don't eroll
in class
SELECT * FROM Students s WHERE s.SID NOT IN (SELECT SID FROM StudentInGroup)
```

Result:


100 %

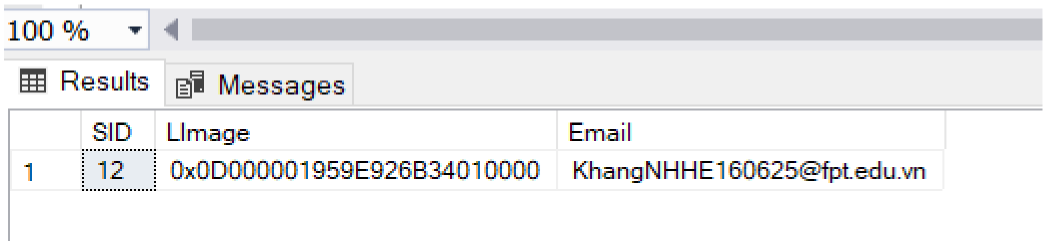
Results | Messages

	SID	LImage	Email
1	3	0x0D000001959E926B34010000	LanNNHE150254@fpt.edu.vn
2	14	0x0D000001959E926B34010000	HungNVHE161049@fpt.edu.vn
3	15	0x0D000001959E926B34010000	TungNDHE161613@fpt.edu.vn
4	16	0x0D000001959E926B34010000	PhucVHHE161615@fpt.edu.vn
5	17	0x0D000001959E926B34010000	QuyenBVHE161625@fpt.edu.vn
6	18	0x0D000001959E926B34010000	MinhPDHE161652@fpt.edu.vn
7	19	0x0D000001959E926B34010000	MinhTNHE161729@fpt.edu.vn
8	20	0x0D000001959E926B34010000	PhucVTHE161744@fpt.edu.vn
9	21	0x0D000001959E926B34010000	AnhPHHE163007@fpt.edu.vn
10	22	0x0D000001959E926B34010000	TuanNMHE163069@fpt.edu.vn
11	23	0x0D000001959E926B34010000	DatBTHE163093@fpt.edu.vn
12	24	0x0D000001959E926B34010000	LamNDHHE163235@fpt.edu.vn
13	25	0x0D000001959E926B34010000	AnhNHHE163241@fpt.edu.vn
14	26	0x0D000001959E926B34010000	AnhNLVHE163350@fpt.edu.vn
15	27	0x0D000001959E926B34010000	BachLXHE164012@fpt.edu.vn
16	28	0x0D000001959E926B34010000	NguyenPDTHE164016@fpt.edu.vn
17	29	0x0D000001959E926B34010000	ex1@fpt.edu.vn
18	30	0x0D000001959E926B34010000	fake@fpt.edu.vn
19	31	0x0D000001959E926B34010000	fake2@fpt.edu.vn

7.

```
--  A query that uses partial matching in the WHERE clause: find the student who absent
in 2022-02-28
SELECT * FROM Students s
WHERE s.SID = (SELECT SID FROM StudentAttendance sa WHERE sa.Status = 0)
```

Result:


100 %

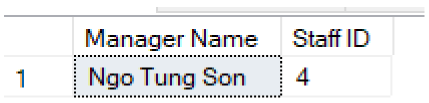
Results | Messages

	SID	LImage	Email
1	12	0x0D000001959E926B34010000	KhangNHHE160625@fpt.edu.vn

8.

```
--  A query that uses a self-JOIN - find the lecturer who is a manager of a nother
lecturer
SELECT ln.SurName + ' ' + ln.MiddleName + ' ' + ln.GivenName as [Manager Name], l2.LID AS
[Staff ID] FROM Lecturers l1 INNER JOIN Lecturers l2
ON l1.LID = l2.ReportTo
INNER JOIN LecturerName ln ON l1.LID = ln.LID
```

Result:



	Manager Name	Staff ID
1	Ngo Tung Son	4

F. The trigger, store procedure and the index should be added:

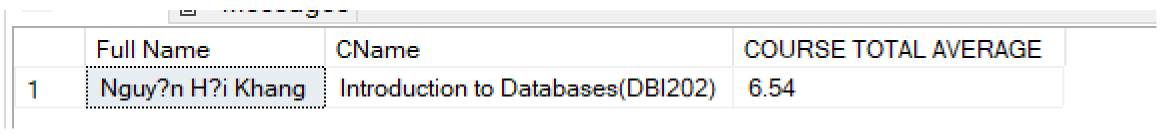
1. Procedure:

```
-- a procedure that show avg of student by Student name and Course Name

DROP PROCEDURE ShowAVGOfStudent
CREATE PROCEDURE ShowAVGOfStudent
@SName varchar(255),
@CName varchar(255)
AS
       SELECT sn.SurName + ' ' + sn.MiddleName + ' ' + sn.GivenName AS [Full Name],
c.CName, SUM(gs.Value * gi.GradeWeight) AS [COURSE TOTAL AVERAGE] FROM Students s
       INNER JOIN GradeOfStudent gs ON s.SID = gs.SID
       INNER JOIN Courses c ON C.CID = gs.CID
       INNER JOIN GradeItems gi ON gi.GrateItemID = gs.GradeItemID
       INNER JOIN StudentName sn ON s.SID = sn.SID
       WHERE sn.GivenName = @SName AND c.CName = @CName
       GROUP BY sn.SurName + ' ' + sn.MiddleName + ' ' + sn.GivenName, c.CName

EXEC ShowAVGOfStudent @SName = 'Khang', @CName = 'Introduction to Databases(DBI202)';
```

Result:

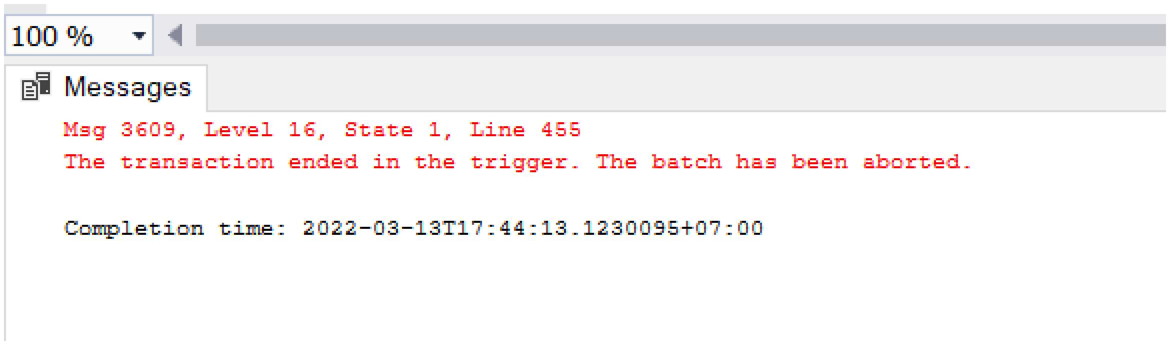


	Full Name	CName	COURSE TOTAL AVERAGE
1	Nguy?n H?i Khang	Introduction to Databases(DBI202)	6.54

2. Trigger:

```
-- Trigger: When a lecturer enter a grade that smaller than 0, the transaction will
rollback
CREATE TRIGGER trigger_validPoint
ON GradeOfStudent
AFTER INSERT
AS
       DECLARE @ValidPoint INT;
       SELECT @ValidPoint = [Value] FROM inserted
       IF @ValidPoint < 0
       BEGIN
              ROLLBACK TRANSACTION
       END
INSERT INTO GradeOfStudent(LID, SID, CID, GradeItemID, [Value]) VALUES (2, 12, 2, 13, -
8);
```

Result:



```
Msg 3609, Level 16, State 1, Line 455
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2022-03-13T17:44:13.1230095+07:00
```

This is Git of this project, you can clone and use them

https://github.com/khangcutetk/AI1604_KhangNH_Assigntment2_Lecturer_Sonnt.git