ASSIGNMENT DBI202

Student: Vu Duy Hung

ID: HE163653

Class: SE1647-DBI SLOT 3

Lecturer: Ngo Tung Son

A. Brief description of the database

This database consists of 7 tables:

1. Student (10 records):
   - 10 record Student have id HE16__
2. Group (1 record):

- SE1647

3. Subject (5 records):
   - DBI202
   - JPA113
   - MAD101
   - PRO192
   - WED201

4. Assessment (25 records):
   - DBI202_ASM, DBI202_FE, DBI202_LAB, DBI202_PE, DBI202_PT
   - JPD113_ASM, JPD113_FE, JPD113_PT1, JPD113_PT2, JPD113_PT3
   - MAD101_ASM, MAD101_FE, MAD101_PT1, MAD101_PT2, MAD101_PT3
   - PRO192_ASM, PRO192_FE, PRO192_LAB, PRO192_PE, PRO192_PT
   - WED201_ASM, WED201_FE, WED201_LAB, WED201_PE, WED201_PT

5. Semester (2 records):
   - Spring 2022
   - Summer 2022

6. Lecture (10 records): 10 records Lecture have id
   - AnhTH
   - HoaNM
   - HungVD
   - HuyNT
   - KhangNM
   - NgocNM
   - QuangTQ
   - SonHT
   - ThoaLT
   - ThuyPT

7. Student_Assessment (250 records) details:

   - HE1610 takes JPD113 (NOT PASS), WED201 (PASS), DBI202 (PASS), PRO192 (PASS), MAD101 (PASS)

   - HE1611 takes JPD113 (PASS), WED201 (PASS), DBI202 (PASS), PRO192 (PASS)

- HE1612 takes JPD113 (PASS), WED201 (PASS), DBI202 (NOT PASS), PRO192 (PASS), MAD101 (PASS)

- HE1613 takes JPD113 (PASS), WED201 (PASS), DBI202 (NOT PASS), PRO192 (PASS), MAD101 (PASS)

- HE1614 takes JPD113 (NOT PASS), WED201 (PASS), DBI202 (NOT PASS), PRO192 (PASS), MAD101 (PASS)

- HE1615 takes JPD113 (PASS), WED201 (PASS), DBI202 (PASS), PRO192 (PASS), MAD101 (PASS)

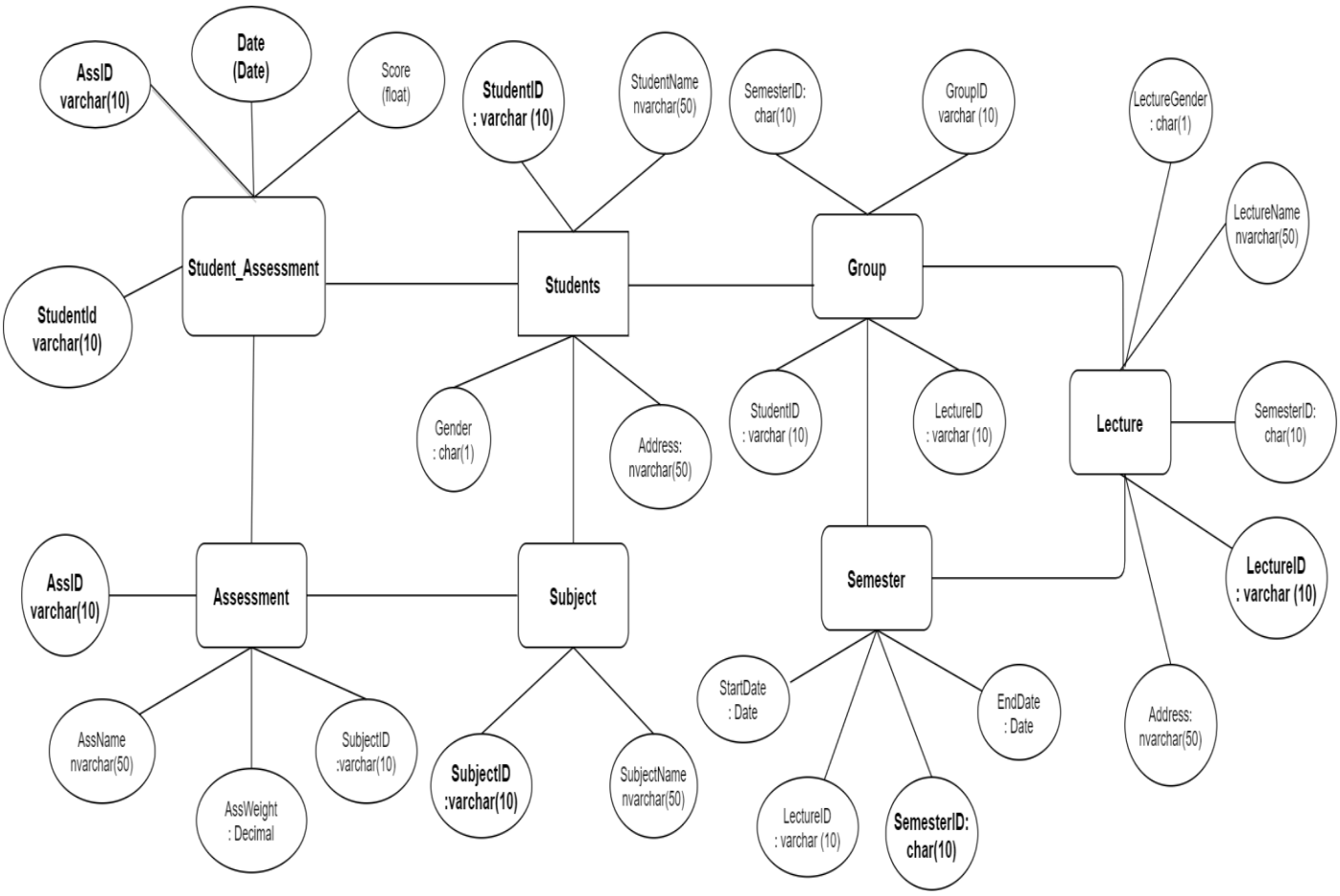
- HE1610 takes JPD113 (NOT PASS), WED201 (PASS), DBI202 (NOT PASS), PRO192 (PASS), MAD101 (PASS)

- HE1617 takes JPD113 (NOT PASS), WED201 (PASS), DBI202 (PASS), PRO192 (PASS), MAD101 (NOT PASS)

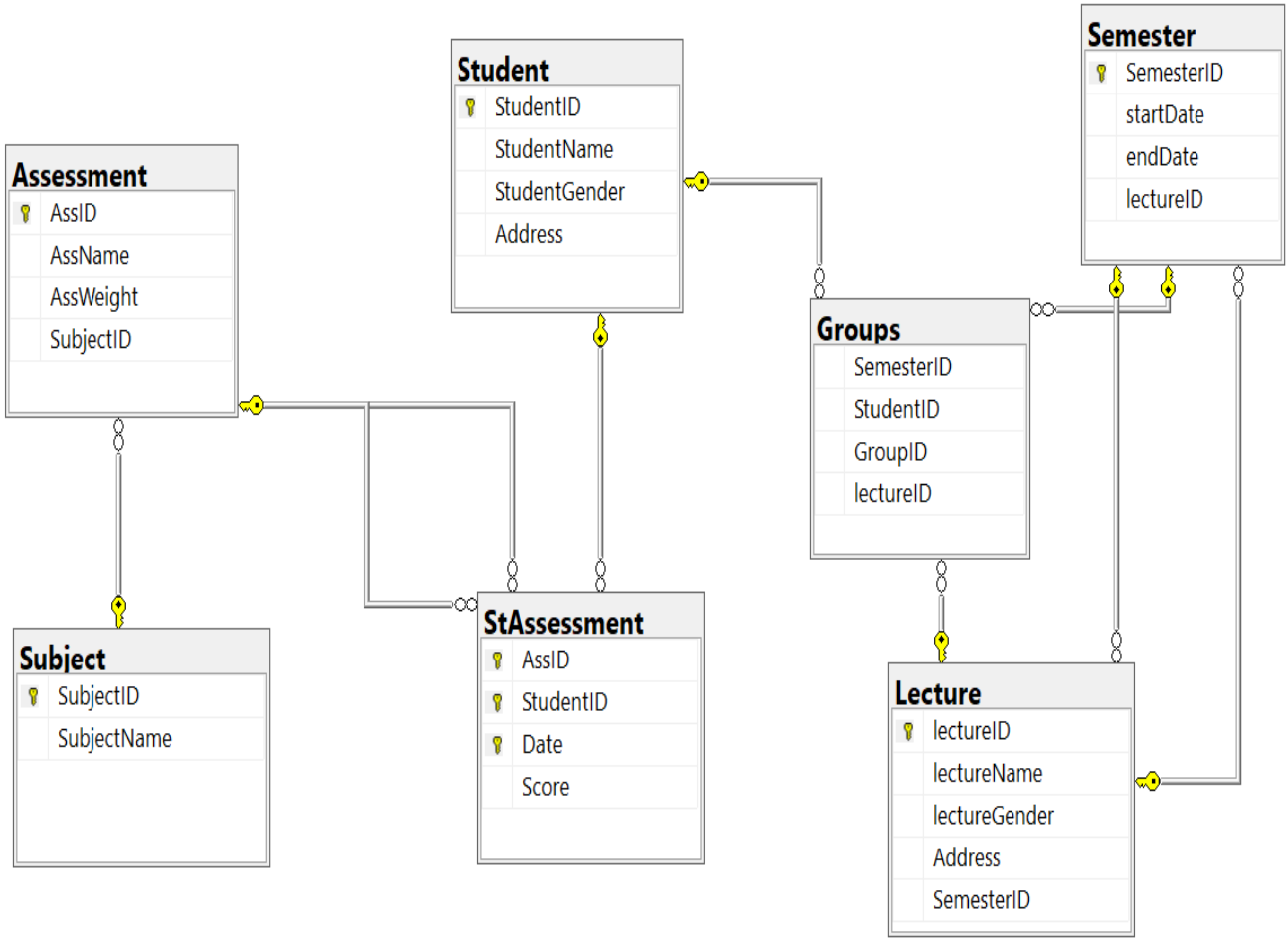
- HE1618 takes JPD113 (PASS), WED201 (PASS), DBI202 (PASS), PRO192 (PASS), MAD101 (PASS)

- HE1619 takes JPD113 (NOT PASS), WED201 (PASS), DBI202 (PASS), PRO192 (PASS), MAD101 (PASS)

B. An ERD that fully describes the database



C. The relational schema derived from the ERD that is at least in 3NF



D. The set of database statements used to create the tables.

1)

```
CREATE TABLE Student (
        [StudentID] [varchar] (10) NOT NULL PRIMARY KEY,
        [StudentName] [nvarchar] (50) NOT NULL,
        [StudentGender] [char] (1) NOT NULL,
        [Address] [nvarchar] (50) NOT NULL)
```

2)

```
CREATE TABLE Subject (
        [SubjectID] [varchar] (10) NOT NULL PRIMARY KEY,
        [SubjectName] [nvarchar] (40) NOT NULL)
```

3)

```
CREATE TABLE Assessment(
        [AssID] [varchar] (10) NOT NULL PRIMARY KEY,
        [AssName] [nvarchar] (50) NOT NULL,
        [AssWeight] [float] NOT NULL,
        [SubjectID] [varchar] (10) NOT NULL FOREIGN KEY REFERENCES
Subject(SubjectID))
```

4)

```
CREATE TABLE StAssessment(
        [AssID] [varchar] (10) NOT NULL FOREIGN KEY REFERENCES
Assessment(AssID),
        [StudentID] [varchar] (10) NOT NULL,
        [Date] [Date] NOT NULL,
        [Score] [float] NOT NULL,
        CONSTRAINT PK_StAssessment PRIMARY KEY (AssID,StudentID,Date),
        CONSTRAINT fk_Student FOREIGN KEY (StudentID) REFERENCES
Student([StudentID]),
        CONSTRAINT fk_Assessment FOREIGN KEY (AssID) REFERENCES
[Assessment](AssID))
```

5)

```
CREATE TABLE Groups (
        [SemesterID] [nvarchar] (50) FOREIGN KEY REFERENCES
Semester(SemesterID),
        [StudentID] [varchar] (10) NOT NULL FOREIGN KEY REFERENCES
Student(StudentID),
        [GroupID] [varchar] (10) NOT NULL,
        [lectureID] [varchar] (10) NOT NULL FOREIGN KEY REFERENCES
Lecture(lectureID),)
```

6)

```
CREATE TABLE Semester(
        [SemesterID] [nvarchar] (50) NOT NULL PRIMARY KEY,
        [startDate] [date] NOT NULL,
        [endDate] [date] NOT NULL,
        [lectureID] [varchar] (10) FOREIGN KEY REFERENCES Lecture(lectureID),)
```

7)

```
CREATE TABLE Lecture(
        [lectureID] [varchar] (10) NOT NULL PRIMARY KEY,
        [lectureName] [nvarchar] (50) NOT NULL,
        [lectureGender] [char] (1) NOT NULL,
        [Address] [nvarchar] (50) NOT NULL,
        [SemesterID] [nvarchar] (50) FOREIGN KEY REFERENCES
Semester(SemesterID),)
```

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

1) **USE FUNCTION, AGGREGATE, SUB-QUERY**:
   +) Students can check their results at the end of semester as following Query

```
SELECT tb1.StudentName, tb1.SubjectName, tb1.StudentID, tb1.SemesterID,
tb1.startDate, tb1.endDate, SUM(AVGTest) AS AVGSubject,
CASE
        WHEN tb1.INFO = 'FAIL' THEN 'NOTPASS'
        WHEN SUM(tb1.AVGTest) < 5 THEN 'NOTPASS'
        ELSE 'PASS'
        END AS [STATUS]
FROM
(SELECT s.StudentName AS StudentName,  s.StudentID AS StudentID,
su.SubjectName AS SubjectName, g.SemesterID, se.startDate, se.endDate,
ISNULL(sa.score,0) AS Grade, ISNULL((sa.score * a.AssWeight),0) AS [AVGTest],
CASE
        WHEN ISNULL(sa.score,0) = 0 THEN 'FAIL'
```

```
        WHEN sa.Score < 4 AND a.AssName = 'Final Exam' THEN 'FAIL'
        ELSE 'PASS'
        END AS [INFO]
FROM  StAssessment sa  LEFT JOIN Assessment a ON a.AssID = sa.AssID
                       LEFT JOIN Subject su  ON su.SubjectID = a.SubjectID
                       LEFT JOIN Student s  ON sa.StudentID = s.StudentID
                       LEFT JOIN Groups g ON g.StudentID = sa.StudentID
                       LEFT JOIN Lecture l ON l.lectureID = g.lectureID
                       LEFT JOIN Semester se ON l.SemesterID = se.SemesterID)
tb1

GROUP BY tb1.StudentName,tb1.SubjectName, tb1.INFO, tb1.StudentID,
tb1.SemesterID,
tb1.startDate, tb1.endDate

ORDER BY tb1.StudentID
```

+) Result:

	StudentName	SubjectName	StudentID	SemesterID	startDate	endDate	AVGSubject	STATUS
1	Nguyen Van A	Japanese	HE1610	Summer 2022	2022-01-01	2022-04-04	3.8	NOTPASS
2	Nguyen Van A	Wed Design	HE1610	Spring 2022	2022-01-01	2022-04-04	6.75	PASS
3	Nguyen Van A	Database	HE1610	Spring 2022	2022-01-01	2022-04-04	6.05	PASS
4	Nguyen Van A	Wed Design	HE1610	Summer 2022	2022-01-01	2022-04-04	6.75	PASS
5	Nguyen Van A	Object-Oriented Programming	HE1610	Spring 2022	2022-01-01	2022-04-04	7.75	PASS

2) **A QUERY THAT USES ORDER BY**

+) Sort the list by StudentID

```
SELECT s.StudentName, s.StudentID, s.StudentGender, s.Address FROM Student s

ORDER BY s.StudentID
```

+) Result:

	StudentName	StudentID	StudentGender	Address
1	Nguyen Van A	HE1610	M	TQ
2	Nguyen Van B	HE1611	M	HN
3	Tran Thi C	HE1612	F	HY
4	Tran Thi D	HE1613	F	HG
5	Tran Van E	HE1614	M	TQ
6	Nguyen Van F	HE1615	M	TQ
7	Hoang Thi G	HE1616	F	HD
8	Cao Van H	HE1617	M	BN
9	Nguyen Van I	HE1618	M	BG
10	Hoang Thi K	HE1619	F	LC

3) A QUERY THAT USES INNER JOINS

+) Join 2 Table Student and Group to display information:

```
SELECT s.StudentID, s.StudentName, g.GroupID, g.SemesterID, g.lectureID
FROM Student s

INNER JOIN Groups g ON s.StudentID = g.StudentID;
```

+) Result:

	StudentID	StudentName	GroupID	SemesterID	lectureID
1	HE1610	Nguyen Van A	SE1647	Spring 2022	AnhTH
2	HE1611	Nguyen Van B	SE1647	Spring 2022	HoaNM
3	HE1612	Tran Thi C	SE1647	Spring 2022	HungVD
4	HE1613	Tran Thi D	SE1647	Spring 2022	HuyNT
5	HE1614	Tran Van E	SE1647	Spring 2022	ThoaLT
6	HE1615	Nguyen Van F	SE1647	Spring 2022	ThuyPT
7	HE1616	Hoang Thi G	SE1647	Spring 2022	KhangNM
8	HE1617	Cao Van H	SE1647	Spring 2022	HoaNM
9	HE1618	Nguyen Van I	SE1647	Spring 2022	HuyNT
10	HE1619	Hoang Thi K	SE1647	Spring 2022	ThoaLT
11	HE1610	Nguyen Van A	SE1647	Summer 2022	AnhTH
12	HE1611	Nguyen Van B	SE1647	Summer 2022	HoaNM
13	HE1612	Tran Thi C	SE1647	Summer 2022	HungVD
14	HE1613	Tran Thi D	SE1647	Summer 2022	HuyNT
15	HE1614	Tran Van E	SE1647	Summer 2022	ThoaLT
16	HE1615	Nguyen Van F	SE1647	Summer 2022	SonHT
17	HE1616	Hoang Thi G	SE1647	Summer 2022	QuangTQ
18	HE1617	Cao Van H	SE1647	Summer 2022	HungVD
19	HE1618	Nguyen Van I	SE1647	Summer 2022	HuyNT
20	HE1619	Hoang Thi K	SE1647	Summer 2022	NgocNM

4) A QUERY THAT USE FUNCTION AGGREGATE, SUB-QUERY, GROUP BY

+) For each subject, student can check their grade average as below Query:

```
SELECT s.StudentName AS StudentName, su.SubjectName AS SubjectName ,
s.StudentID AS StudentID,
SUM(sa.Score * a.AssWeight) AS [AVG]
FROM StAssessment sa LEFT JOIN Assessment a ON a.AssID = sa.AssID
                     LEFT JOIN Subject su ON su.SubjectID = a.SubjectID
                     LEFT JOIN Student s ON sa.StudentID = s.StudentID

GROUP BY su.SubjectName, s.StudentName, s.StudentID
```

+) Result:

	StudentName	SubjectName	StudentID	AVG
1	Nguyen Van A	Japanese	HE1610	3.8
2	Nguyen Van A	Database	HE1610	6.05
3	Nguyen Van A	Discrete mathematics	HE1610	7.65
4	Nguyen Van A	Object-Oriented Programming	HE1610	7.75
5	Nguyen Van A	Wed Design	HE1610	6.75
6	Nguyen Van B	Japanese	HE1611	8
7	Nguyen Van B	Database	HE1611	7.4
8	Nguyen Van B	Discrete mathematics	HE1611	6.97
9	Nguyen Van B	Object-Oriented Programming	HE1611	5.96
10	Nguyen Van B	Wed Design	HE1611	7.025

5) **A QUERY THAT USE FUNCTION AGGREGATE, SUB-QUERY, GROUP BY, LEFT JOIN**

+) Students can check their GPA at the end of semester as following Query:

```
SELECT tb1.StudentName, AVG([AVG]) AS GPA FROM
(SELECT s.StudentName AS StudentName, su.SubjectName AS SubjectName ,
SUM(sa.Score * a.AssWeight) AS [AVG]
FROM StAssessment sa  LEFT JOIN Assessment a ON a.AssID = sa.AssID
                      LEFT JOIN Subject su ON su.SubjectID = a.SubjectID
                      LEFT JOIN Student s ON sa.StudentID = s.StudentID
GROUP BY su.SubjectName, s.StudentName) tb1

GROUP BY tb1.StudentName
```

+) Result:

	StudentName	StudentID	GPA
1	Nguyen Van A	HE1610	6.4
2	Nguyen Van B	HE1611	7.071
3	Tran Thi C	HE1612	6.415
4	Tran Thi D	HE1613	6.605
5	Tran Van E	HE1614	5.955
6	Nguyen Van F	HE1615	6.81
7	Hoang Thi G	HE1616	6.205
8	Cao Van H	HE1617	5.595
9	Nguyen Van I	HE1618	6.825
10	Hoang Thi K	HE1619	6.325

6) **A QUERY THAT USE SELF JOIN, WHERE CLAUSE**

+) To select couple with the some address:

```
SELECT a.StudentName AS StudentName1, b.StudentName AS StudentName2,
a.Address
FROM Student a, Student b
WHERE a.StudentID <> b.StudentID
AND a.Address = b.Address
ORDER BY a.Address;
```

+)Result

	StudentName1	StudentName2	Address
1	Nguyen Van A	Tran Van E	TQ
2	Nguyen Van A	Nguyen Van F	TQ
3	Tran Van E	Nguyen Van A	TQ
4	Tran Van E	Nguyen Van F	TQ
5	Nguyen Van F	Nguyen Van A	TQ
6	Nguyen Van F	Tran Van E	TQ

7) **A QUERY THAT USE GROUP BY AND HAVING CLAUSE**

+) To check their Number of Score

```
SELECT Sts.StudentID, Sts.Score, COUNT(Sts.Score) AS NumberOfScore
FROM StAssessment Sts
INNER JOIN Student s ON s.StudentID = Sts.StudentID
GROUP BY Sts.StudentID, Sts.Score HAVING COUNT(Sts.Score) = 5

ORDER BY Sts.StudentID
```

+) Result:

	StudentID	Score	NumberOfScore
1	HE1612	7	5
2	HE1613	7	5
3	HE1617	5	5
4	HE1617	7	5
5	HE1619	7	5

8) A QUERY THAT USE INNER JOIN AND ORDER BY

+) Join 3 Table Group, Lecture and Semester to display information:

```
SELECT  g.GroupID, l.lectureID, s.SemesterID, s.startDate, s.endDate
FROM ((Groups g
INNER JOIN Lecture l ON l.lectureID   = g.lectureID)
INNER JOIN Semester s ON g.SemesterID = s.SemesterID)

ORDER BY SemesterID
```

+) Result:

	GroupID	lectureID	SemesterID	startDate	endDate
1	SE1647	AnhTH	Spring 2022	2022-01-01	2022-04-04
2	SE1647	HoaNM	Spring 2022	2022-01-01	2022-04-04
3	SE1647	HungVD	Spring 2022	2022-01-01	2022-04-04
4	SE1647	HuyNT	Spring 2022	2022-01-01	2022-04-04
5	SE1647	ThoaLT	Spring 2022	2022-01-01	2022-04-04
6	SE1647	ThuyPT	Spring 2022	2022-01-01	2022-04-04
7	SE1647	KhangNM	Spring 2022	2022-01-01	2022-04-04
8	SE1647	HoaNM	Spring 2022	2022-01-01	2022-04-04
9	SE1647	HuyNT	Spring 2022	2022-01-01	2022-04-04
10	SE1647	ThoaLT	Spring 2022	2022-01-01	2022-04-04
11	SE1647	AnhTH	Summer 2022	2022-05-05	2022-09-09
12	SE1647	HoaNM	Summer 2022	2022-05-05	2022-09-09
13	SE1647	HungVD	Summer 2022	2022-05-05	2022-09-09
14	SE1647	HuyNT	Summer 2022	2022-05-05	2022-09-09
15	SE1647	ThoaLT	Summer 2022	2022-05-05	2022-09-09
16	SE1647	SonHT	Summer 2022	2022-05-05	2022-09-09
17	SE1647	QuangTQ	Summer 2022	2022-05-05	2022-09-09
18	SE1647	HungVD	Summer 2022	2022-05-05	2022-09-09
19	SE1647	HuyNT	Summer 2022	2022-05-05	2022-09-09
20	SE1647	NgocNM	Summer 2022	2022-05-05	2022-09-09

9) A QUERY THAT USE WHERE CLAUSE

+) Search for Student with Score between 0 and 5

```
SELECT s.StudentName, s.StudentID, sts.Score, sts.AssID, sts.Date
FROM Student s
INNER JOIN StAssessment sts ON s.StudentID = sts.StudentID
AND sts.Score BETWEEN 0 AND 5
WHERE sts.AssID = 'DBI202_ASM'
```

```
GROUP BY s.StudentID, s.StudentName, sts.Score, sts.AssID, sts.Date

ORDER BY s.StudentID
```

+)Result

	StudentName	StudentID	Score	AssID	Date
1	Nguyen Van B	HE1611	3	DBI202_ASM	2022-02-15
2	Tran Van E	HE1614	1	DBI202_ASM	2022-02-15
3	Hoang Thi G	HE1616	4	DBI202_ASM	2022-02-15
4	Hoang Thi K	HE1619	4	DBI202_ASM	2022-02-15

10) A QUERY THAT USE AGGREGATE FUNCTION

+) Student can check their Ranking at the end of semester as following Query:

```
SELECT  s.StudentName, s.StudentID,
   CASE
      WHEN AVG(Score) <= 5 THEN 'Poor'
      WHEN AVG(Score) > 7 AND AVG(Score) <= 9 THEN 'WelDone'
      ELSE 'Excellent'
   END AS [Ranking]
FROM StAssessment St
JOIN Student s On s.StudentID = St.StudentID

GROUP BY s.StudentID, s.StudentName
```

+) Result:

	StudentName	StudentID	Ranking
1	Nguyen Van A	HE1610	Excellent
2	Nguyen Van B	HE1611	WelDone
3	Tran Thi C	HE1612	Excellent
4	Tran Thi D	HE1613	Excellent
5	Tran Van E	HE1614	Excellent
6	Nguyen Van F	HE1615	Excellent
7	Hoang Thi G	HE1616	Excellent
8	Cao Van H	HE1617	Excellent
9	Nguyen Van I	HE1618	Excellent
10	Hoang Thi K	HE1619	Excellent

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

1. **USE STORE PROCEDURE**

   +)To Query Score as the below Query:

```
GO
CREATE PROCEDURE uspFindScore
(
	@min_score AS DECIMAL = 9,
	@max_score AS DECIMAL = 10,
	@name AS VARCHAR(max)
)
AS
BEGIN
	SELECT
		s.StudentID, s.StudentName, st.AssID, st.Score, st.Date
	FROM
		Student s, StAssessment st
	WHERE
		Score > @min_score AND
		Score < @max_score AND
		s.StudentName LIKE @name + '%'
	ORDER BY
		s.StudentID
END;

EXEC uspFindScore

@name = 'N';
```

+) Result

	StudentID	StudentName	AssID	Score	Date
1	HE1610	Nguyen Van A	DBI202_PT	9.5	2022-01-15
2	HE1610	Nguyen Van A	JPD113_ASM	9.5	2022-03-20
3	HE1610	Nguyen Van A	JPD113_PT1	9.5	2022-01-15
4	HE1610	Nguyen Van A	JPD113_PT1	9.5	2022-01-15
5	HE1610	Nguyen Van A	JPD113_PT3	9.5	2022-03-15
6	HE1610	Nguyen Van A	MAD101_ASM	9.5	2022-03-20
7	HE1610	Nguyen Van A	MAD101_FE	9.5	2022-04-01
8	HE1610	Nguyen Van A	MAD101_PT1	9.5	2022-01-15
9	HE1610	Nguyen Van A	PRO192_ASM	9.5	2022-02-15
10	HE1610	Nguyen Van A	PRO192_FE	9.5	2022-04-01
11	HE1610	Nguyen Van A	PRO192_LAB	9.5	2022-03-15
12	HE1611	Nguyen Van B	DBI202_PT	9.5	2022-01-15
13	HE1611	Nguyen Van B	JPD113_ASM	9.5	2022-03-20
14	HE1611	Nguyen Van B	JPD113_PT1	9.5	2022-01-15
15	HE1611	Nguyen Van B	JPD113_PT1	9.5	2022-01-15
16	HE1611	Nguyen Van B	JPD113_PT3	9.5	2022-03-15
17	HE1611	Nguyen Van B	MAD101_ASM	9.5	2022-03-20
18	HE1611	Nguyen Van B	MAD101_FE	9.5	2022-04-01
19	HE1611	Nguyen Van B	MAD101_PT1	9.5	2022-01-15
20	HE1611	Nguyen Van B	PRO192_ASM	9.5	2022-02-15

2. USE TRIGGER

+)To check valid of GroupID, Trigger as below query :

```
GO
CREATE TRIGGER trigger_asm ON Groups
AFTER INSERT,UPDATE,DELETE
AS
	DECLARE @GID nvarchar(10);
	DECLARE @COUNT nvarchar(10);
	SELECT @GID = GroupID  FROM inserted;
	SELECT @COUNT = COUNT(*) FROM Groups WHERE GroupID = @GID;
	IF @COUNT > 2
	BEGIN
		PRINT 'Existed';
		ROLLBACK TRAN

	END
```

TEST CASE 1:

```
INSERT INTO Groups VALUES ('Spring 2022', 'HE1610', 'SE1647', 'AnhTH')
```

+) Result:

```
Existed
Msg 3609, Level 16, State 1, Line 112
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2022-07-17T22:16:12.1322219+07:00
```

TEST CASE 1:

```
INSERT INTO Groups VALUES ('Spring 2022', 'HE1610', 'SE1648', 'AnhTH')
```

+) Result:

```
(1 row affected)

Completion time: 2022-07-17T22:33:31.5164051+07:00
```

3. USE INDEX

+)An index do not allow any duplicate values to be inserted into the table

```
CREATE INDEX index_id

ON Student (StudentID)
```