

1. CREATE DATABASE University;

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'Databases' node under 'DESKTOP-UP37904\SQLEXPRESS (SQ)' is expanded, showing various system databases and other databases like AP, Examples, lab6, ProductOrders, and University. In the center pane, a query window titled 'SQLQuery1.sql - DESKTOP-UP37904\owner (53)* - Microsoft SQL Server Management Studio' contains the following SQL command:

```
--Lichen Liang  
--create database with name University  
CREATE DATABASE University;
```

The status bar at the bottom right indicates 'Ready'. The message pane shows 'Commands completed successfully.' and the completion time: 'Completion time: 2020-03-09T23:51:56.5232202-04:00'. The status bar also shows 'DESKTOP-UP37904\SQLEXPRESS ... DESKTOP-UP37904\owner ... master 00:00:01 0 rows'.

2. USE University;

CREATE TABLE Students

(StudentID	int	NOT NULL PRIMARY KEY IDENTITY,
FirstName	varchar(50)	NOT NULL,
LastName	varchar(50)	NOT NULL,
Phone	varchar(50)	NULL);

CREATE TABLE Courses

(CourseID	int	NOT NULL PRIMARY KEY IDENTITY,
CourseName	varchar(90)	NOT NULL,
CoursePrice	money	NOT NULL DEFAULT 0 CHECK (CoursePrice >= 0));

CREATE TABLE StudyGroups

(StudentID	int	REFERENCES Students(StudentID),
CourseID	int	REFERENCES Courses(CourseID));

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'University' database is selected. In the center pane, a query window displays the following SQL code:

```

--Lichen Liang
--Within the University database
USE University;

/*For the students table, the studentID is set to be the identity primary key because it needs to be unique for each student.
StudentID, FirstName, and LastName should be non-null because a person must have a full name and its corresponding unique student ID number.
Otherwise, phone number would be incomplete in the database.
However, phone number can be NULL since a person is allowed to have no phone for any kind of reason.
If the Phone is set to NOT NULL and the person does not have a phone number, then we have to put some wrong data in the database to satisfy the constraint.
*/
CREATE TABLE Students
(
    StudentID int NOT NULL PRIMARY KEY IDENTITY,
    FirstName varchar(50) NOT NULL,
    LastName varchar(50) NOT NULL,
    Phone varchar(50) NULL
);

/*The CourseID is set to be identity primary key because every course is unique and should have a unique ID number.
For every course, its CourseID, CourseName, and CoursePrice must be NOT NULL since it must have a ID number, a name, and a price, which can be set to 0 instead of NULL.
The CoursePrice is also set to default 0, which means free, but cannot be negative otherwise it means losing money.
*/
CREATE TABLE Courses
(
    CourseID int NOT NULL PRIMARY KEY IDENTITY,
    CourseName varchar(50) NOT NULL,
    CoursePrice money NOT NULL DEFAULT 0 CHECK (CoursePrice >= 0)
);

/*This is the linking table for Students and Courses.
It has two foreign keys CourseID and StudentsID.
*/
CREATE TABLE StudyGroups
(
    StudentID int REFERENCES Students(StudentID),
    CourseID int REFERENCES Courses(CourseID)
);

```

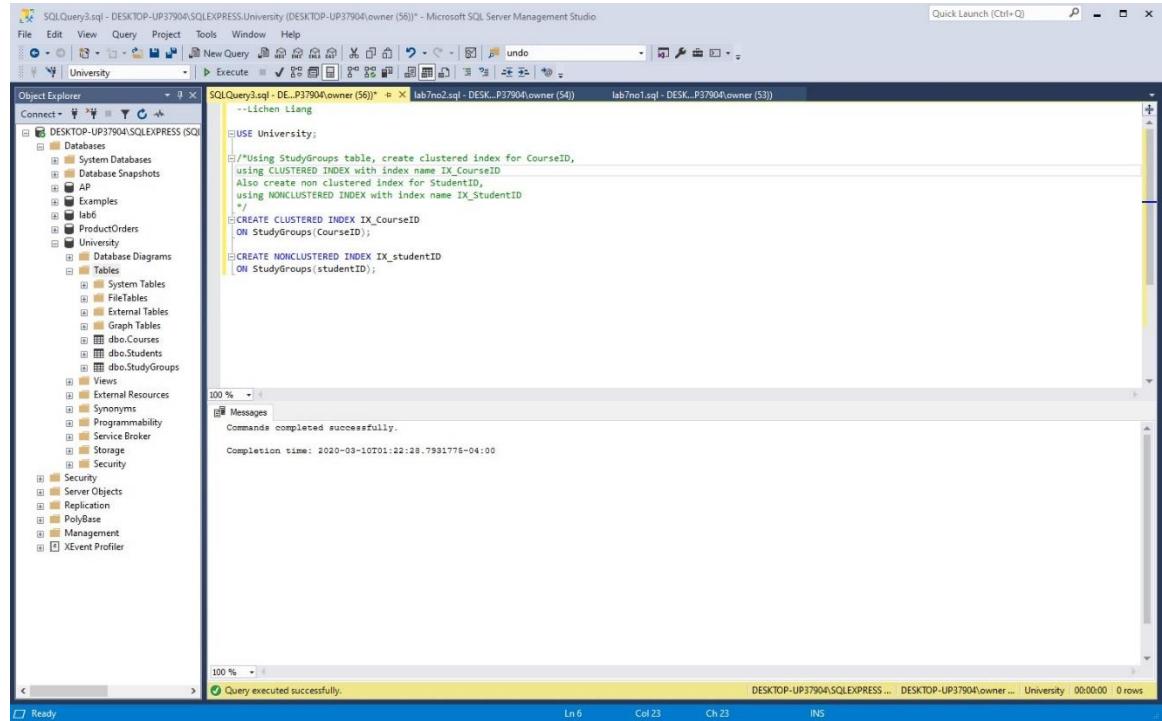
The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2020-03-10T00:59:54.4097499-04:00".

Students table and Courses table have a many-to-many relationship. StudyGroups is their linking table. So both Students and Courses are a one-to-many relationship with the linking table.

3. USE University;

```
CREATE CLUSTERED INDEX IX_CourseID  
ON StudyGroups(CourseID);
```

```
CREATE NONCLUSTERED INDEX IX_studentID  
ON StudyGroups(studentID);
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'University'. The 'Scripting' tab is selected in the toolbar. A query window titled 'SQLQuery3.sql' contains the following T-SQL code:

```
--Lichen Liang  
USE University;  
/*Using StudyGroups table, create clustered index for CourseID,  
using CLUSTERED INDEX with index name IX_CourseID  
Also create non clustered index for StudentID,  
using NONCLUSTERED INDEX with index name IX_StudentID  
*/  
CREATE CLUSTERED INDEX IX_CourseID  
ON StudyGroups(CourseID);  
CREATE NONCLUSTERED INDEX IX_studentID  
ON StudyGroups(studentID);
```

The 'Messages' pane at the bottom displays the output:

```
Commands completed successfully.  
Completion time: 2020-03-10T01:22:28.7931776-04:00
```

The status bar at the bottom right indicates: DESKTOP-UP37904:SQLEXPRESS... - DESKTOP-UP37904\owner... University 00:00:00 0 rows.

4. USE University;

```
ALTER TABLE Students  
ADD FeesPaid bit NOT NULL DEFAULT 0;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'University' database which contains tables like 'Courses', 'Students', and 'StudyGroups'. The central pane displays the T-SQL code:

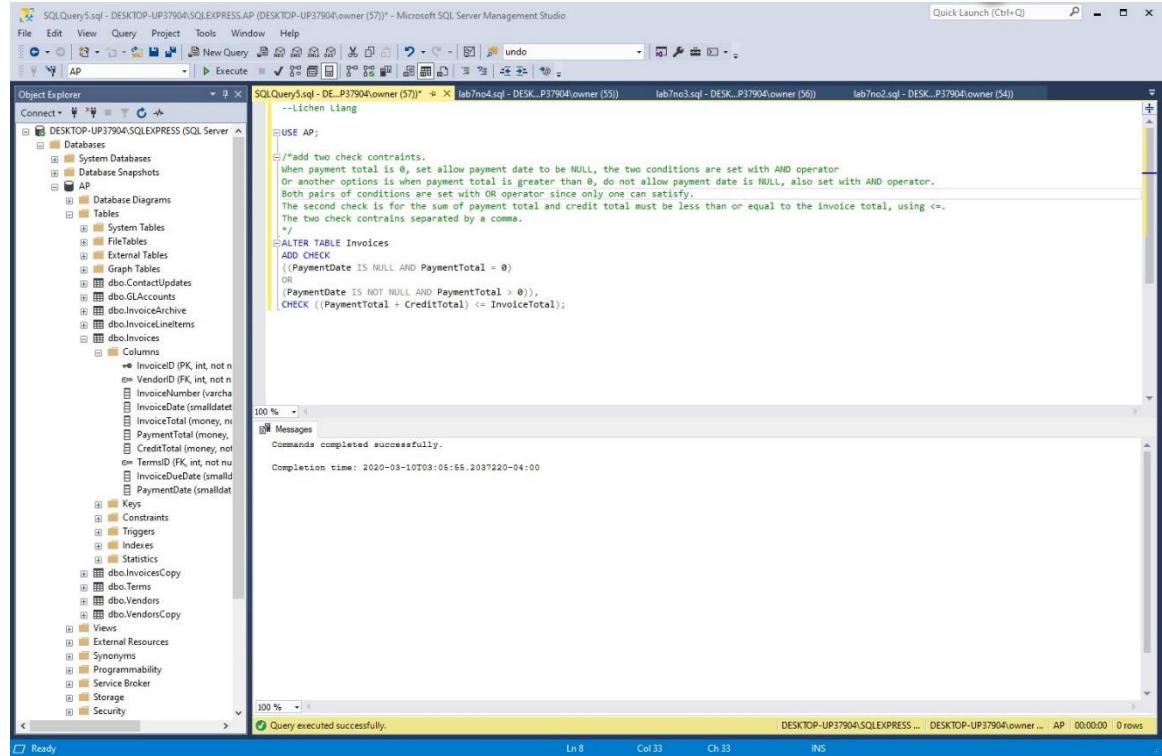
```
--Lichen Liang  
USE University;  
/*Add a column to students table using ALTER TABLE Students ADD,  
with column name FeesPaid, datatype bit(1 for true or 0 for false)  
Cannot be NULL because it's either paid or unpaid(true or false)  
with default set to 0(false)  
*/  
ALTER TABLE Students  
ADD FeesPaid bit NOT NULL DEFAULT 0;
```

The 'Messages' pane at the bottom right shows the command completed successfully with a completion time of 2020-03-10T01:31:12.0092273-04:00.

At the bottom of the screen, a status bar indicates 'Ready', 'En 5', 'Col 65', 'Ch 65', and 'INS'.

5. USE AP;

```
ALTER TABLE Invoices
ADD CHECK
((PaymentDate IS NULL AND PaymentTotal = 0)
OR
(PaymentDate IS NOT NULL AND PaymentTotal > 0)),
CHECK ((PaymentTotal + CreditTotal) <= InvoiceTotal);
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the AP schema which contains tables like Invoices, GLAccounts, and InvoiceLineItems. The central pane displays the T-SQL code for altering the Invoices table:

```
--Lichen Liang
USE AP;
/*add two check constraints.
when payment total is 0, set allow payment date to be NULL, the two conditions are set with AND operator
Or another options is when payment total is greater than 0, do not allow payment date is NULL, also set with AND operator.
Both two conditions are set with OR operator since only one can satisfy.
The second check is for the sum of payment total and credit total must be less than or equal to the invoice total, using <=.
The two check constraints separated by a comma.
*/
ALTER TABLE Invoices
ADD CHECK
((PaymentDate IS NULL AND PaymentTotal = 0)
OR
(PaymentDate IS NOT NULL AND PaymentTotal > 0)),
CHECK ((PaymentTotal + CreditTotal) <= InvoiceTotal);
```

The status bar at the bottom indicates the command was completed successfully.

6. USE University;

DROP TABLE StudyGroups;

CREATE TABLE StudyGroups

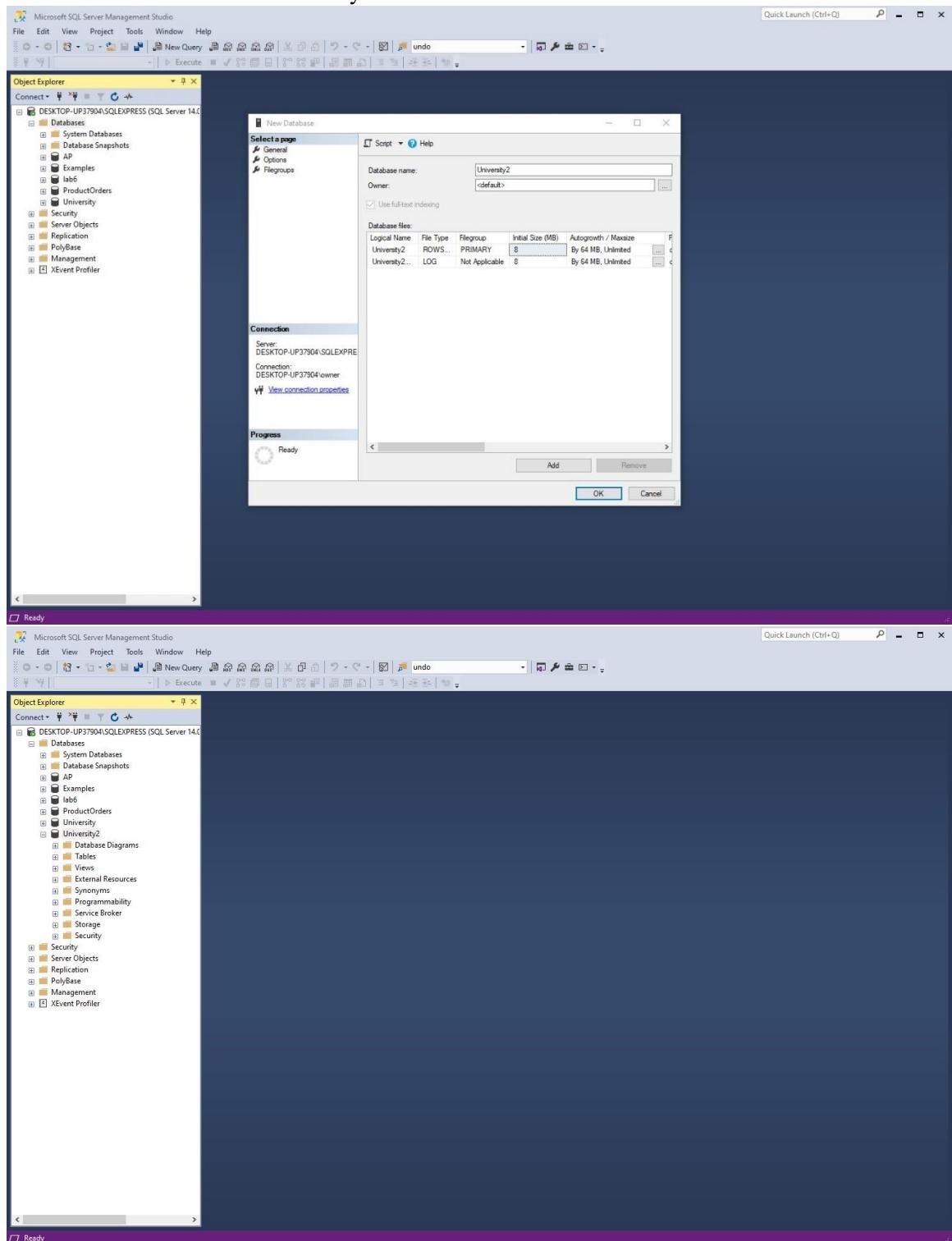
```
(StudentID      int      REFERENCES Students(StudentID),
CourseID       int      REFERENCES Courses(CourseID),
UNIQUE(StudentID, CourseID));
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'University'. The 'Tables' node under 'dbo' contains 'StudyGroups'. The 'Script' node for 'StudyGroups' is selected, displaying the T-SQL code for creating the table. The 'Messages' pane at the bottom right shows the execution results: 'Commands completed successfully.' and the completion time '2020-03-10T03:16:44.0174952-04:00'. The status bar at the bottom indicates 'Query executed successfully.'

```
--USE University;
--Deletes the table named StudyGroups, using DROP TABLE
DROP TABLE StudyGroups;

--re-create the table
--using the UNIQUE to prevent student in the same course twice
CREATE TABLE StudyGroups
(
    StudentID int REFERENCES Students(StudentID),
    CourseID  int REFERENCES Courses(CourseID),
    UNIQUE(StudentID, CourseID)
);
```

7. /*CREATE DATABASE University2*/



University2 database is created.

Remarks

This lab we practiced how to use DDL to manipulate the database, such as CREATE, DROP, ALTER. We also practiced using constraints and understand why they are important for different kind of columns. I think this a very good practice for the lecture as well as the project coming up. A more complicated challenge would be creating a database with more tables, columns, constraints, etc.