**University:**

* **We have to create Data Base for University.**

**The Entities**: Student, Teacher, Branch, Class, Subject, Exam, Faculty, Term, Year, Certificate.

1. **User requirements:** are statements in a natural language plus diagrams of what services the system is expected to provide t system users and the constraints under which it must operate written for customer. Register new student, update student's information and enter teacher's information....

**System requirements: it is most** important to build local database, must has a high security, the system speed and strong and add users and privileges.

**The Attributes:**

Student: stud\_id, stud\_name, stud\_address, stud\_phone.

Teacher: teacher\_id, teacher\_name, teacher\_address, teacher\_phone.

Branch: bran\_id, bran\_name, bran\_address.

Class: class\_id, class\_name.

Subject: sub\_id, sub\_name, sub\_length.

Exam: exam\_id, exam\_name, exam\_date,exam\_time.

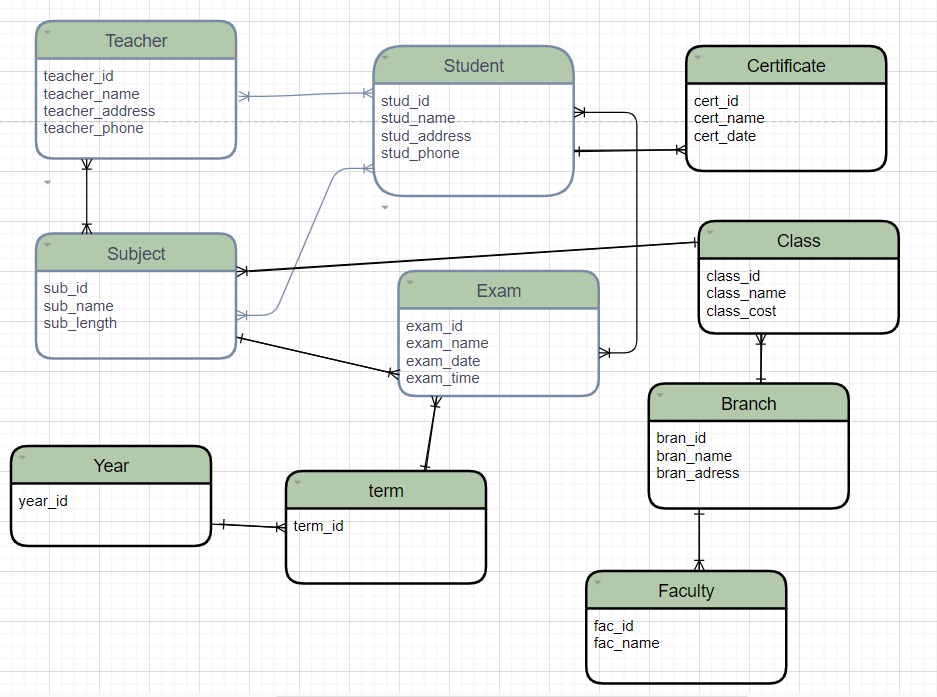
Faculty: fac\_id,fac\_name.

Term: term\_id.

Year: year\_num.

Certificate: cert\_id, cert\_name, cert\_date.

**This is the EAR Diagram:** (by using ERD.drawio)



**The Relationships between Entities are:**

- Has 5 branches.

- There are 4-6 classes in each branch.

- There are 30 students in each class, each student studying one subject at least and takes one or more exams and gets a mark.

- Each class has almost 7-8 subjects.

- Each teacher teaches one or more subjects and each subject teaches by one or more than one teacher.

- There are 4 terms in a year and two tests are conducted in each term.

* **Logical Design:** is an abstract conceptual design. You are not dealing with the physical implementation details; you are merely dealing with identifying the types of information that you need. This logical design process involves arranging all the data into a series of logical relationships called entities andattributes.

- I checked all fields names, and there is no duplication of them.

### - we have to choose the primary key and the foreign key which produces from one-to-many relation and from many to many.

Student

|  |  |  |  |
| --- | --- | --- | --- |
| **Stud\_id** | Stud\_name | Stud\_address | Stud\_phone |

Teacher

|  |  |  |  |
| --- | --- | --- | --- |
| **Teacher\_id** | Teacher\_name | Teacher\_address | Teacher\_phone |

Stud\_teacher

|  |  |  |
| --- | --- | --- |
| Stud\_id  ---------- | Teacher\_id  --------------- | Session\_time |

Subject

|  |  |  |  |
| --- | --- | --- | --- |
| **Sub\_id** | Sub\_name | Sub\_length | Class\_id  ----------- |

Stud\_sub

|  |  |  |
| --- | --- | --- |
| Stud\_id  ----------- | Sub\_id  ---------- | Status |

Teacher\_sub

|  |  |  |
| --- | --- | --- |
| Teacher\_id  --------------- | Sub\_id  --------- | Sub\_date |

Certificate

|  |  |  |  |
| --- | --- | --- | --- |
| **Cert\_id** | Cert\_name | Cert\_date | Stud\_id  ----------- |

Branch

|  |  |  |
| --- | --- | --- |
| **Bran\_id** | Bran\_name | Bran\_address |

Class

|  |  |  |
| --- | --- | --- |
| **Class\_id** | Class\_name | Class\_cost |

Exam

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exam\_id** | Exam\_name | Exam\_date | Exam\_time | Sub\_id  --------- | Term\_id  ----------- |

Stud\_exam

|  |  |  |
| --- | --- | --- |
| Stud\_id  ----------- | Exam\_id  ----------- | mark |

Faculty

|  |  |  |
| --- | --- | --- |
| **Fac\_id** | Fac\_name | Bran\_id  ----------- |

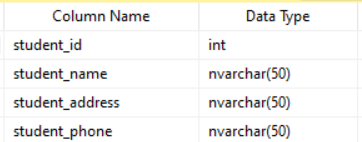
Term

|  |  |
| --- | --- |
| **Term\_id** | Year\_id  ---------- |

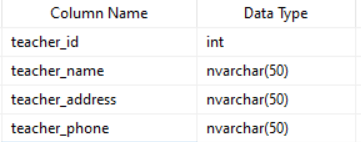
Year

|  |
| --- |
| **Year\_id** |

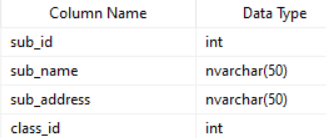
\* We have **Data validation** for each table that is:

Student:  


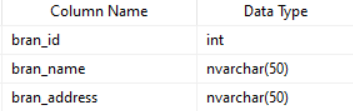
Teacher:



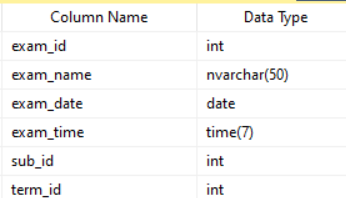
Subject:



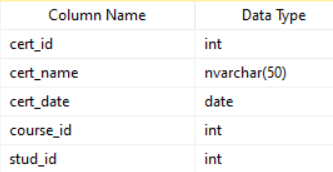
Branch:



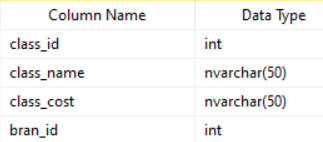
Exam:



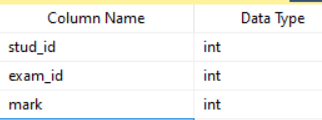
Certificate:



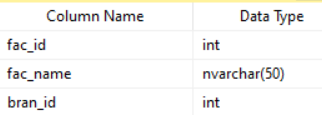
Class:



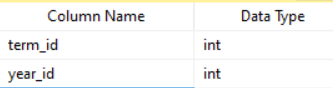
Stud\_exam:



Faculty:



Term:



\* Also, we must take care about **Data Normalization** which is:

A systematic way to ensure that the structure of a database is general-purpose fit and that it is free from any unwanted attributes - such as insertion, update, and accidental deletion - that could lead to data loss. Normalization rules divide larger tables into smaller tables and relate them using relationships. And it has three types:

**First Normalization Form**: that means a table must not contain composite or multi-valued attributes.

**Second Normalization Form:** the data must be in First Normalization Form, and a relation must not contain any partial dependencies (There is no non-key field associated (functionally followed) for part of the key).

**Third Normalization Form:** the data must be in Second Normalization Form and a relation must not contain any transitive functional dependencies on the primary key.

- In this Data Base:

**1NF**: is approved because each field in all tables has single data and do not have any repeated data.

**2NF**: is approved because There is no non-key field associated (functionally followed) for part of the key.

**3NF**: is approved because There are no non-key fields that are associated to other non-key fields.

* In our Data Base we used the Logical Design which provided us with an easy-to-use standard user interface, check functionality, which verifies that user input is complete, correct and reasonable, meaningful data selection, central authorization check for database access and good performance for read access while retaining the hierarchical data display determined by the application logic. And used Normalization which allowed us to eliminate unnecessary duplication of data, increase data consistency, data mapped closely to real-world entities, and help us get the application database structure the right first time. Also, we used the Data Validation which It is cost effective because it saves the right amount of time and money by collecting data sets. It is easy to use and compatible processes as it removes duplication from the complete data set. Data validation can directly help improve the business through improved information gathering, and it consists of an efficient data building that provides standard database and clean dataset information.

We had created all tables in this Data Base:

Create table student:

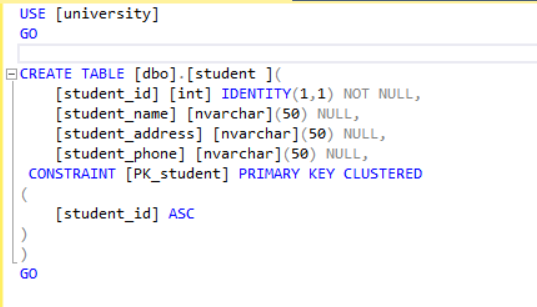
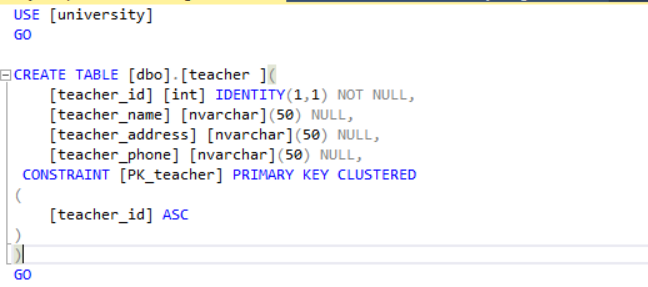
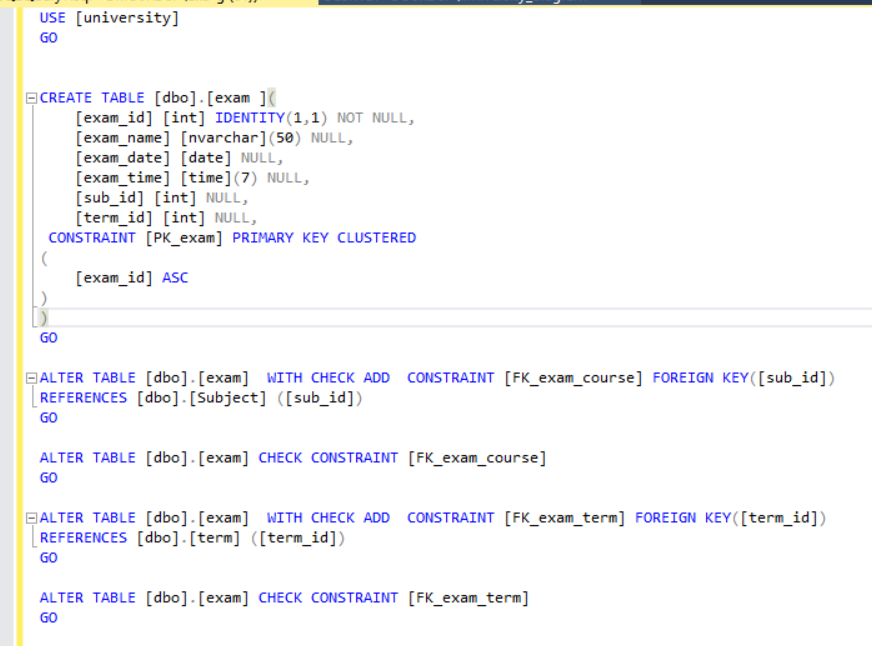


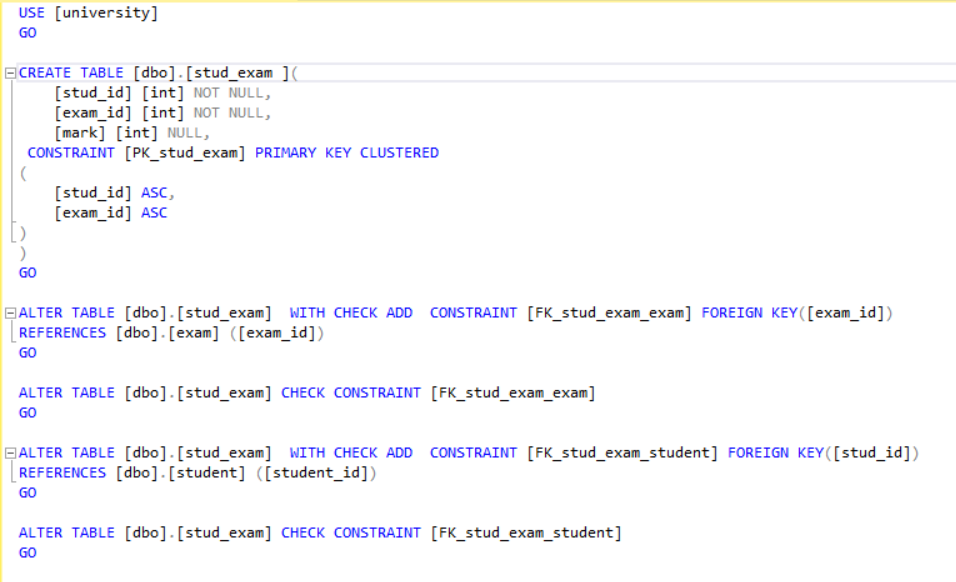
Table teacher:



Exam table:



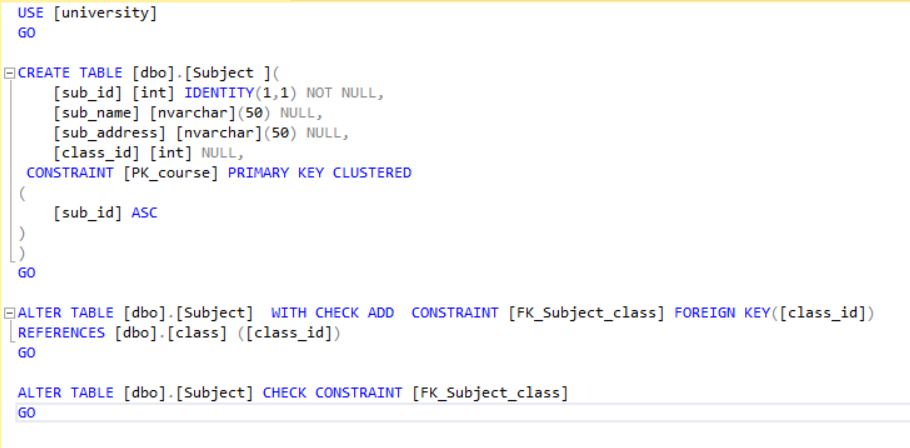
Stud\_exam table:



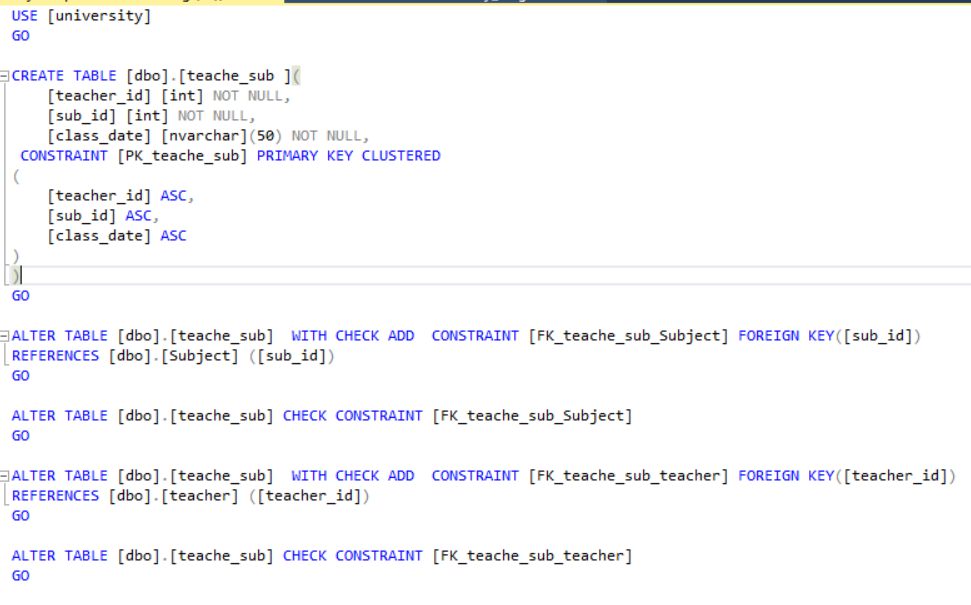
Stud\_teacher table:



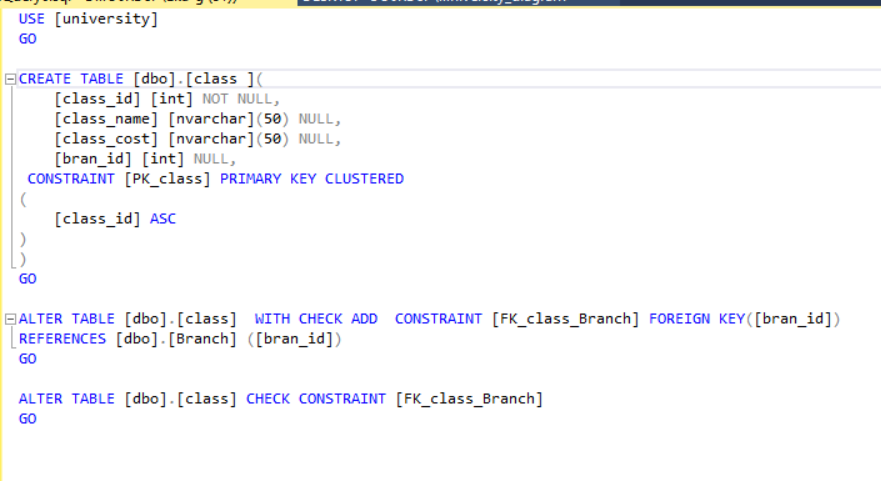
Subject table:



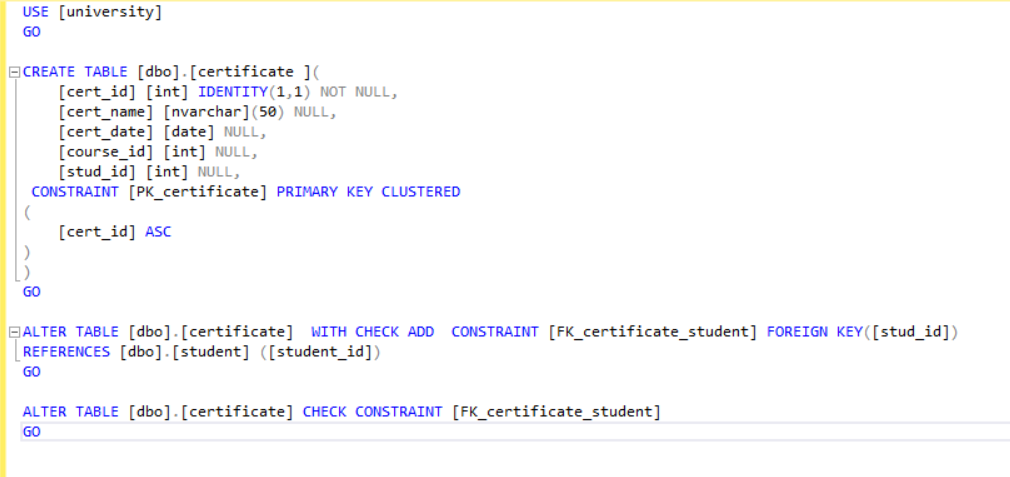
Teacher\_sub table:



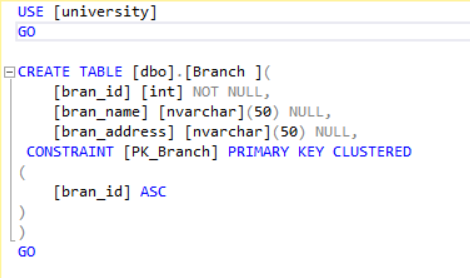
Class table:



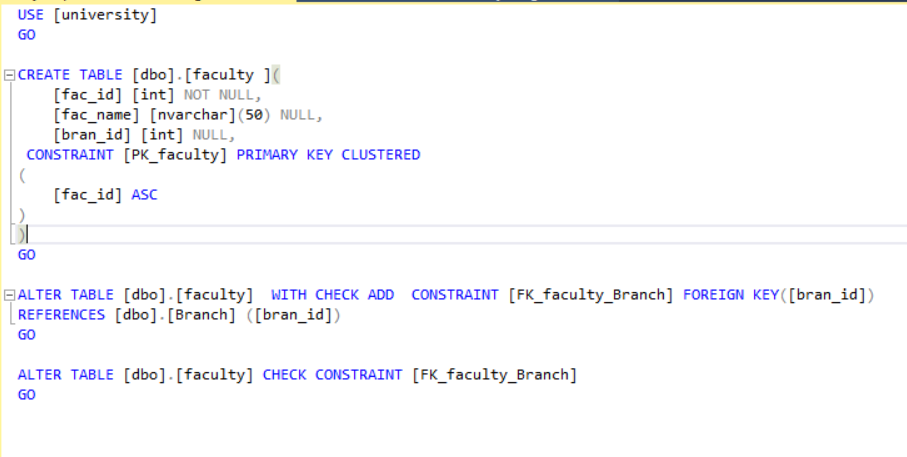
Certificate table:



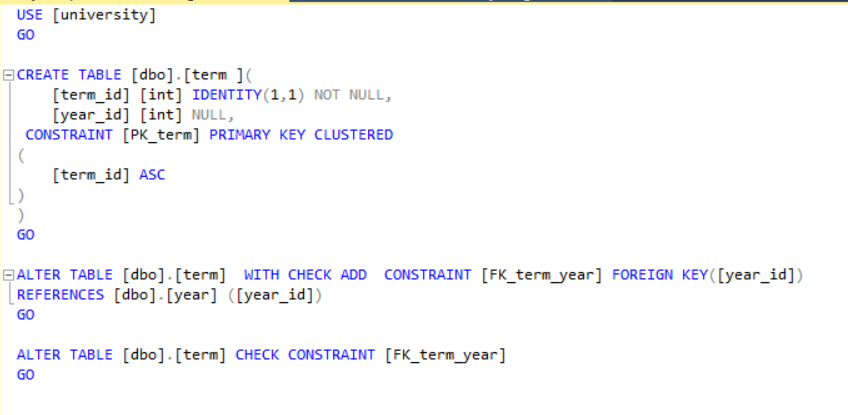
Branch table:



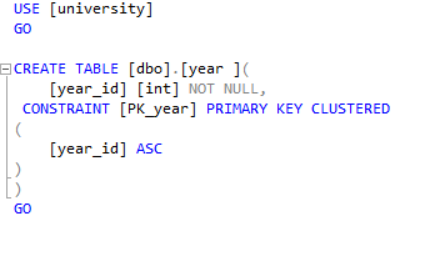
Faculty table:



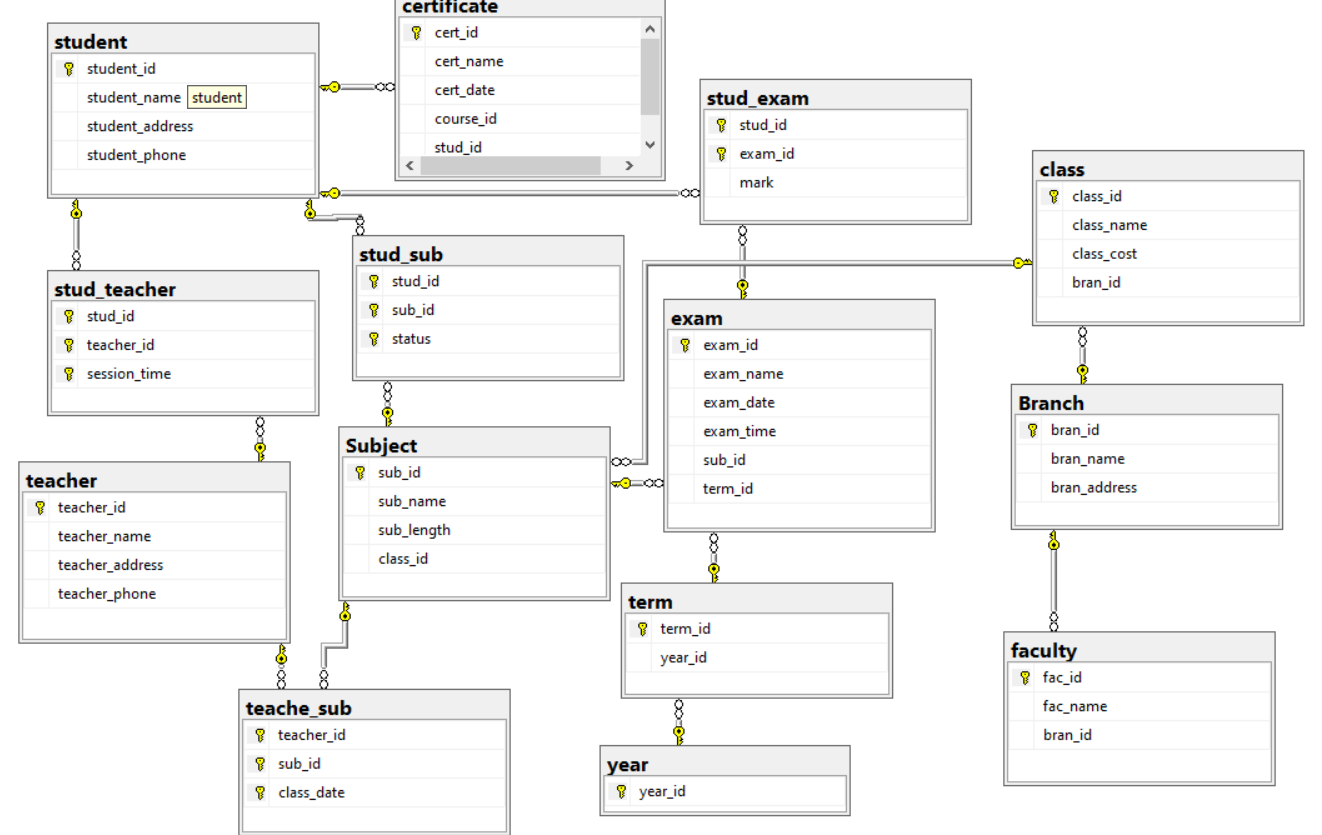
Term table:



Year table:

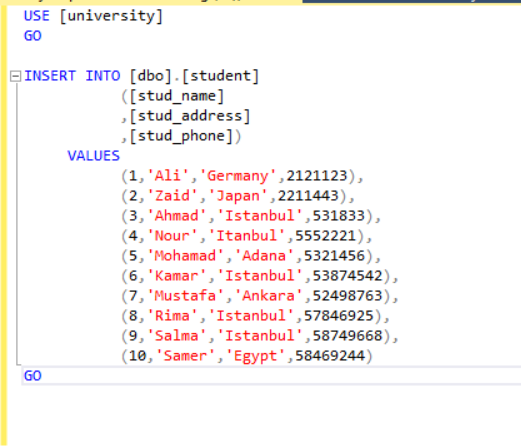


After create all tables the Database Diagram on sql server become:

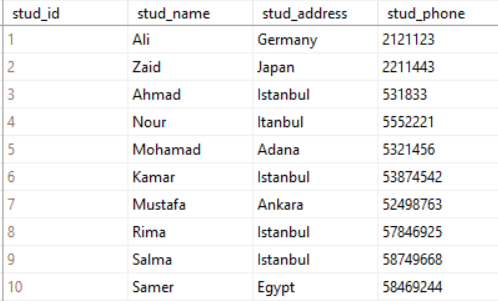


- we have to insert data into the tables by using insert code.

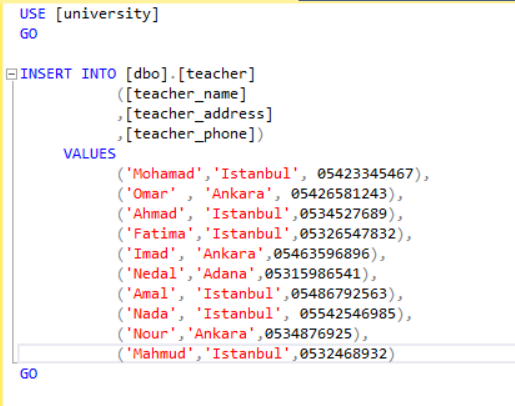
Insert data into student's table:



And the result will be:



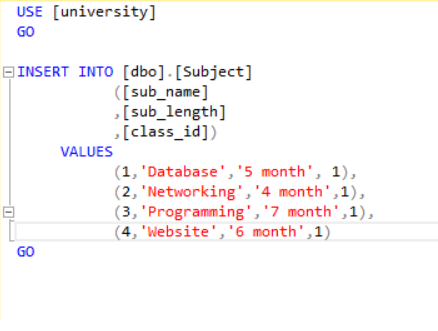
Insert data into teacher's table:



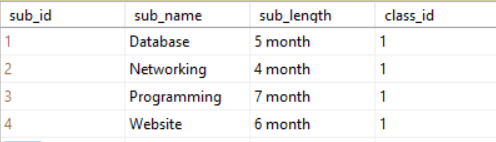
And the result will be:



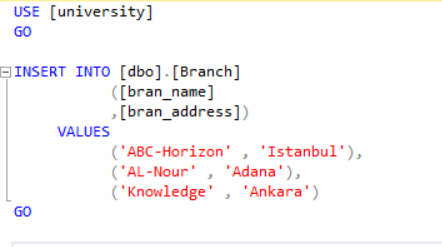
Insert data into Subject's table:



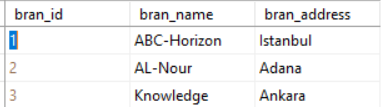
And the result will be:



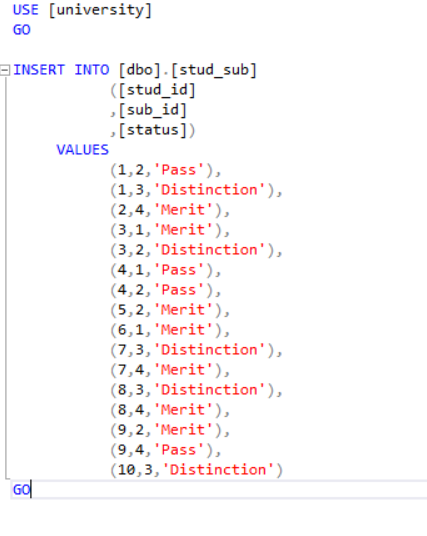
Insert data into Branch's table:



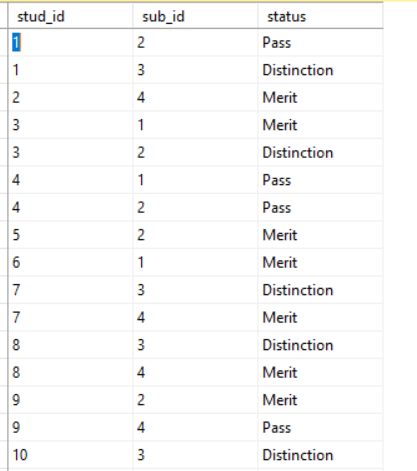
And the result will be:



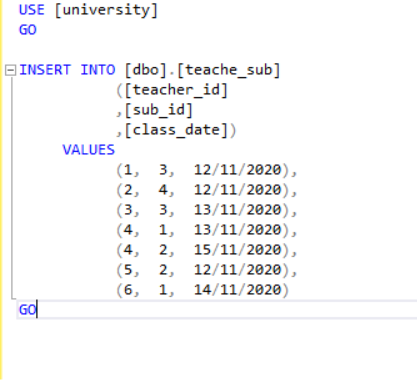
Insert data into stud\_sub's table:



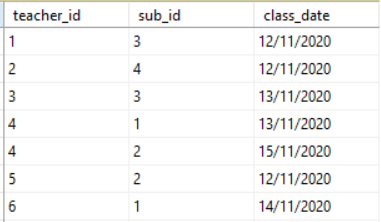
And the result will be:



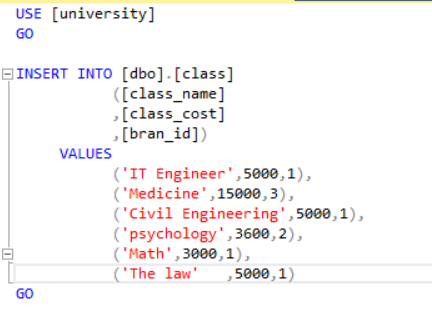
Insert data into teacher\_sub's table:



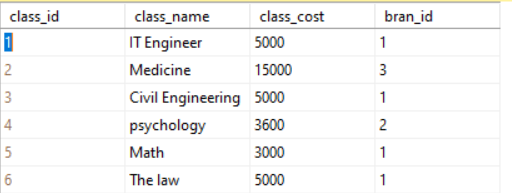
And the result will be:



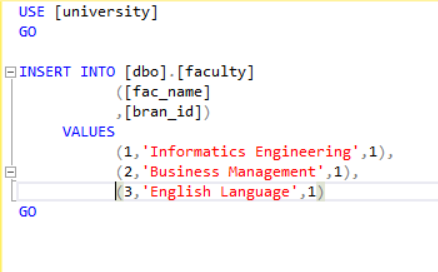
Insert data into class's table:



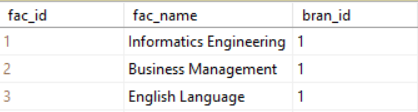
And the result will be:



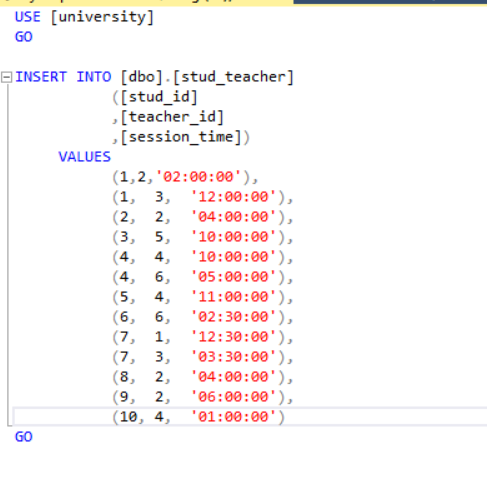
Insert data into faculty's table:



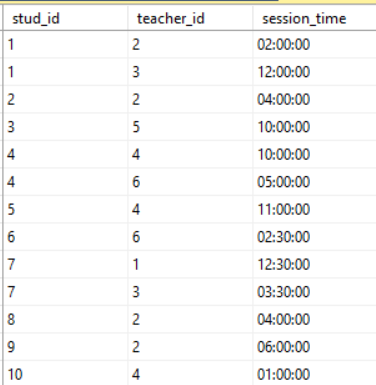
And the result will be:



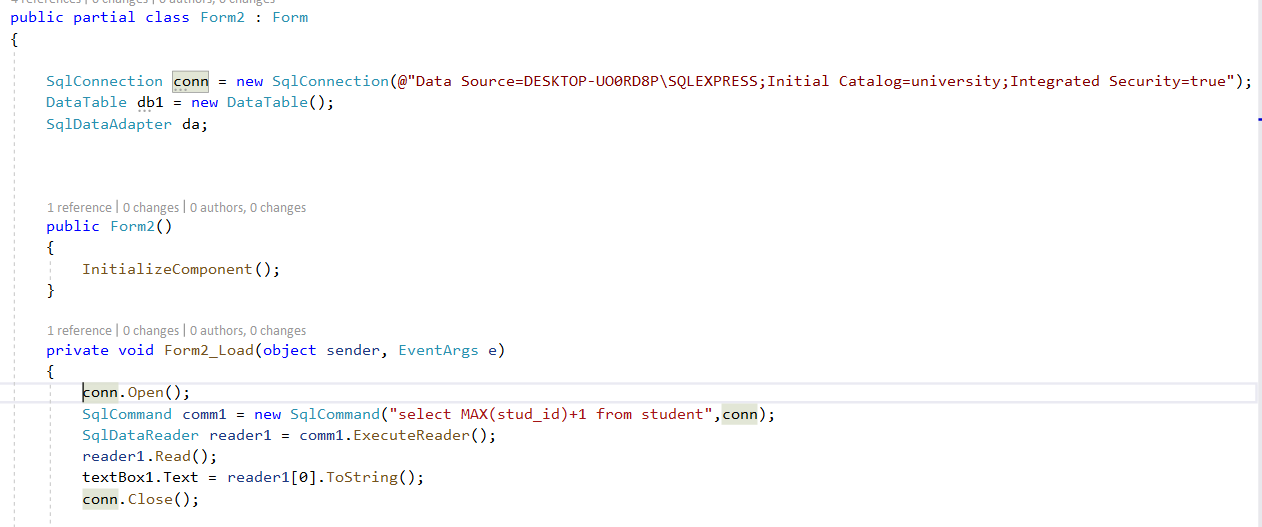
Insert data into stud\_teacher's table:

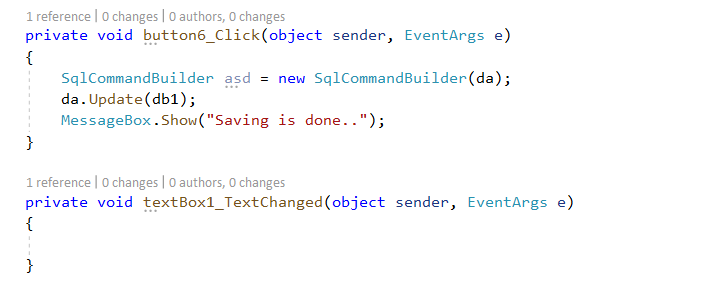
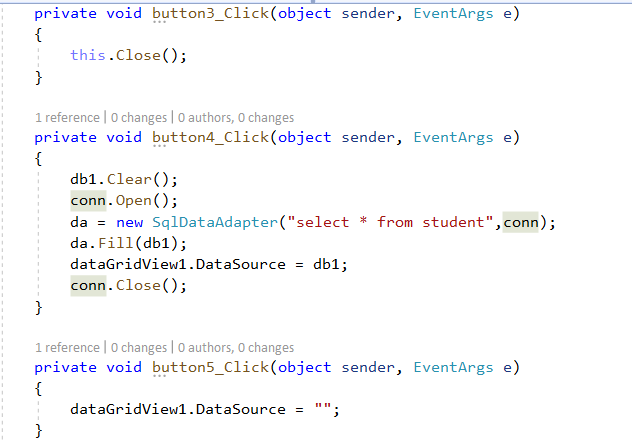


And the result will be:



**Student Interface code:**





**Queries:**

- This query shows all the subject in which each student is registered and his status in each subject:

use university

select student.stud\_name,Subject.sub\_name,stud\_sub.status

from student,Subject,stud\_sub

where student.stud\_id=stud\_sub.stud\_id and Subject.sub\_id=stud\_sub.sub\_id

- This query shows Each teacher with his subject, length of time, and date of each class:

use university

select teacher.teacher\_name,Subject.sub\_name,Subject.sub\_length,teache\_sub.class\_date

from teacher,Subject,teache\_sub

where teacher.teacher\_id=teache\_sub.teacher\_id and Subject.sub\_id=teache\_sub.sub\_id

This query shows Each branch and all its faculties, and all classes in the faculty and its cost:

use university

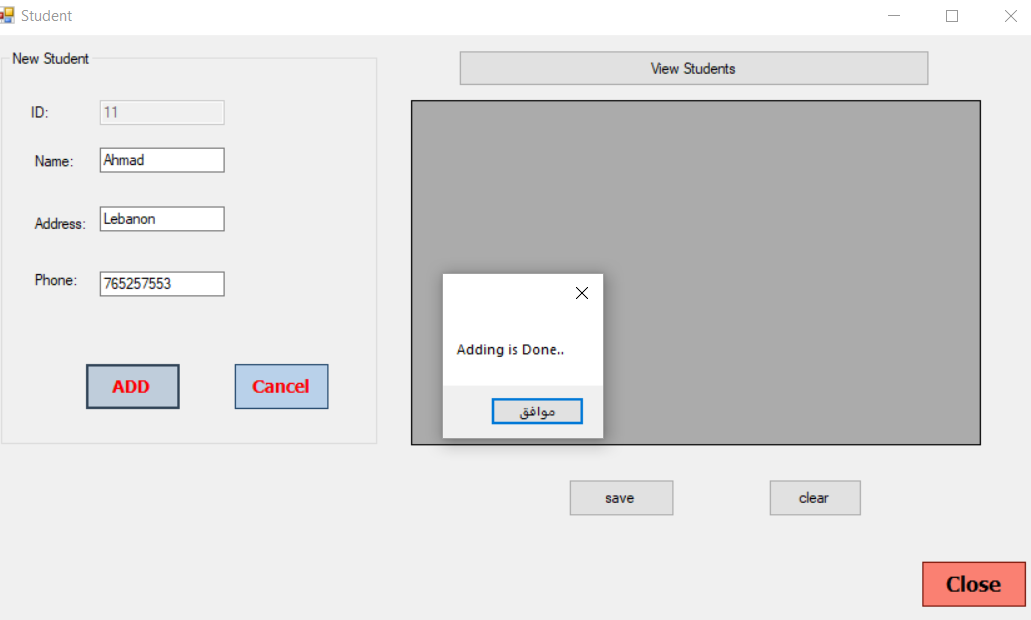
select Branch.bran\_name,faculty.fac\_name,class.class\_name,class.class\_cost

from Branch,faculty,class

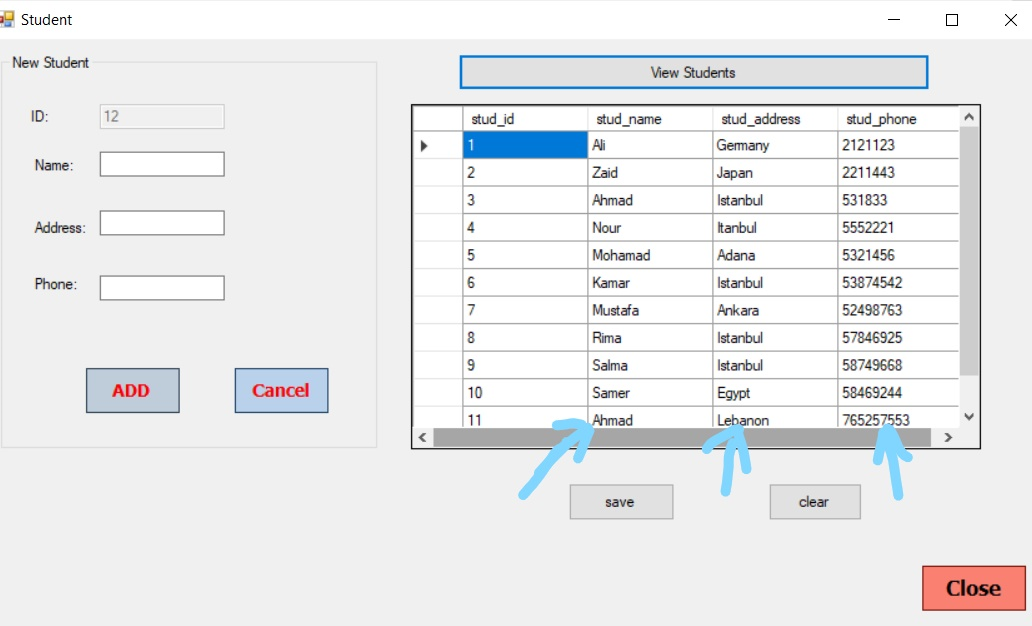
Where Branch.bran\_id = faculty.bran\_id and Branch.bran\_id = class.bran\_id

Test our application:

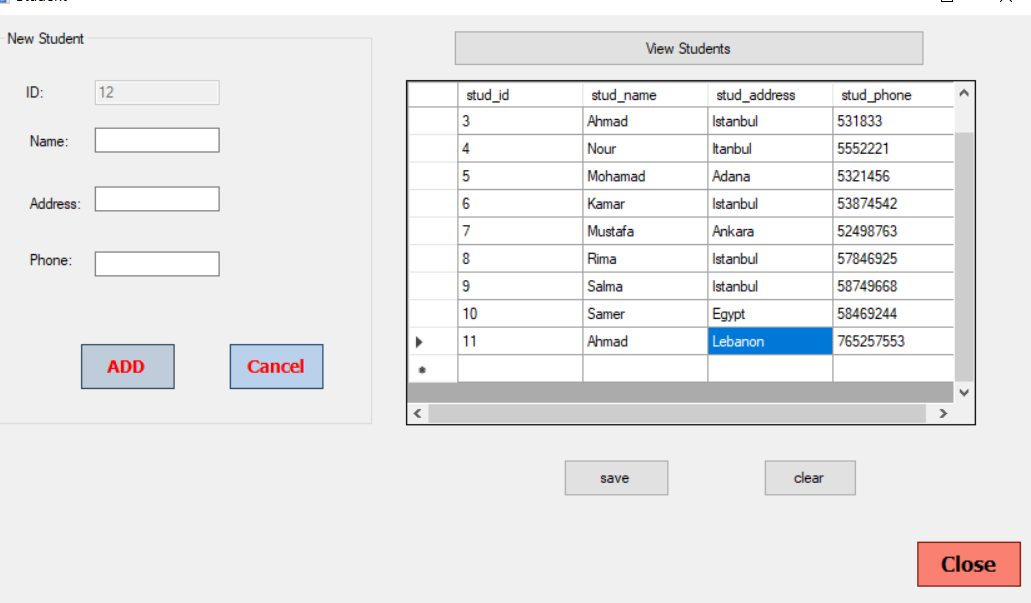
Now I will add a new student ‘Ahmad’:



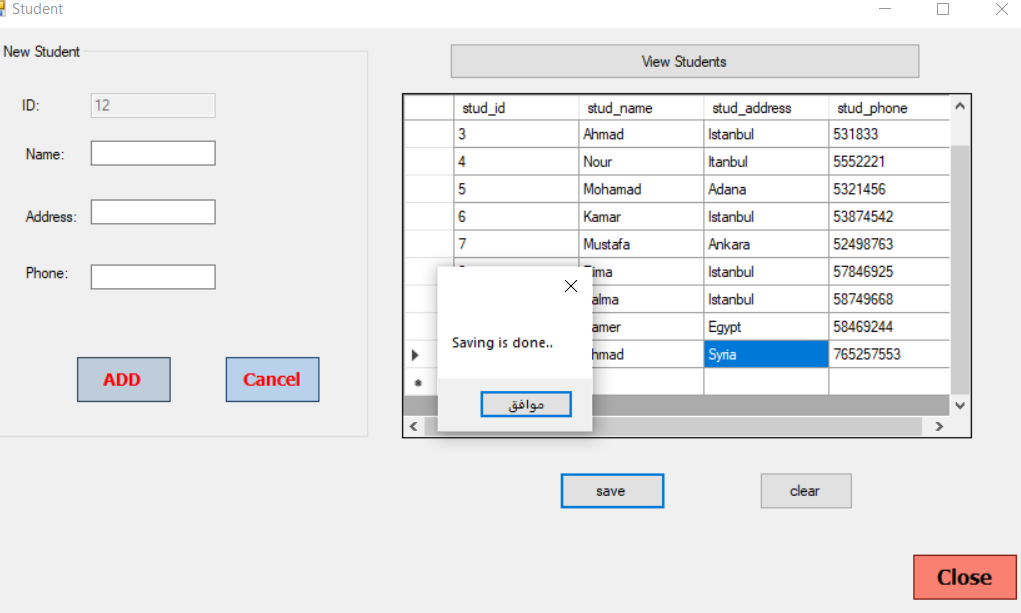
Show the adding is done:



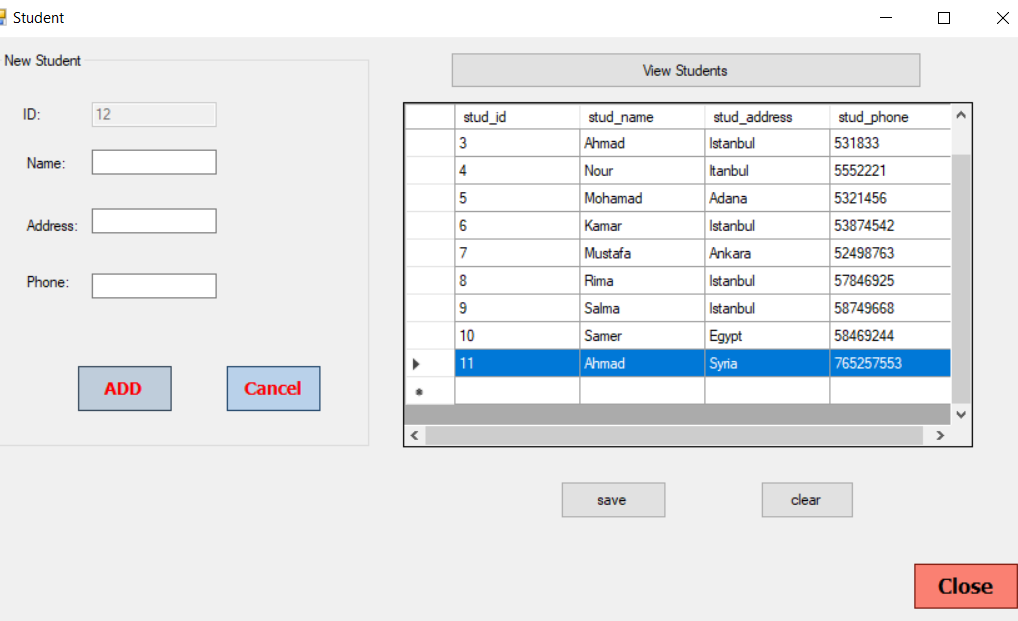
Edit student:



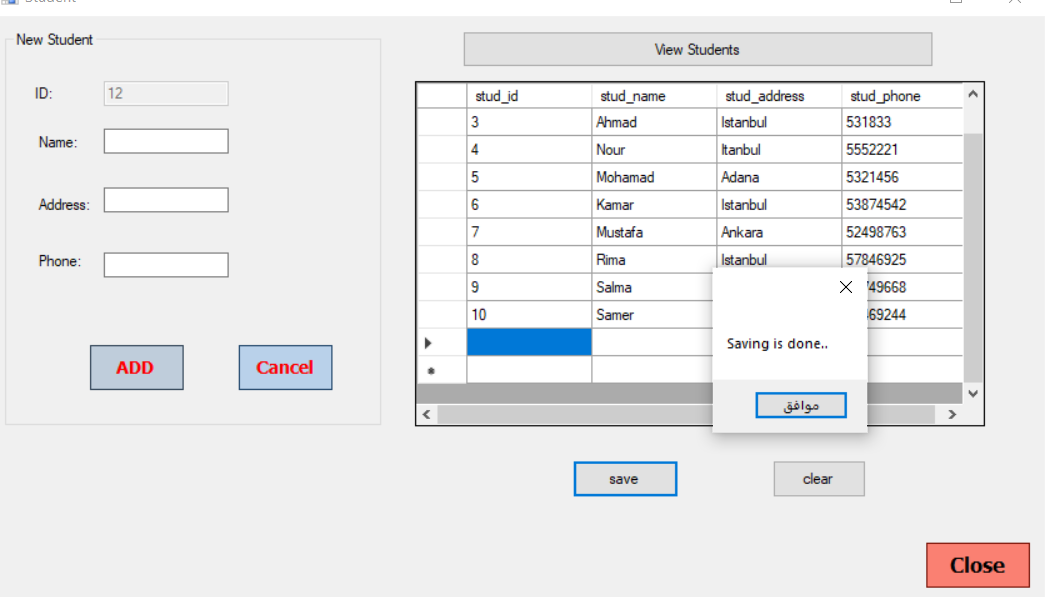
Editing is done:



Delete student:



Deletion is done



**Technical documentation:**

* **We have to create Data Base for University.**

**The Entities**: Student, Teacher, Branch, Class, Subject, Exam, Faculty, Term, Year, Certificate.

1. **User requirements:** are statements in a natural language plus diagrams of what services the system is expected to provide t system users and the constraints under which it must operate written for customer. Register new student, update student's information and enter teacher's information.

**The Attributes:**

Student: stud\_id, stud\_name, stud\_address, stud\_phone.

Teacher: teacher\_id, teacher\_name, teacher\_address, teacher\_phone.

Branch: bran\_id, bran\_name, bran\_address.

Class: class\_id, class\_name.

Subject: sub\_id, sub\_name, sub\_length.

Exam: exam\_id, exam\_name, exam\_date,exam\_time.

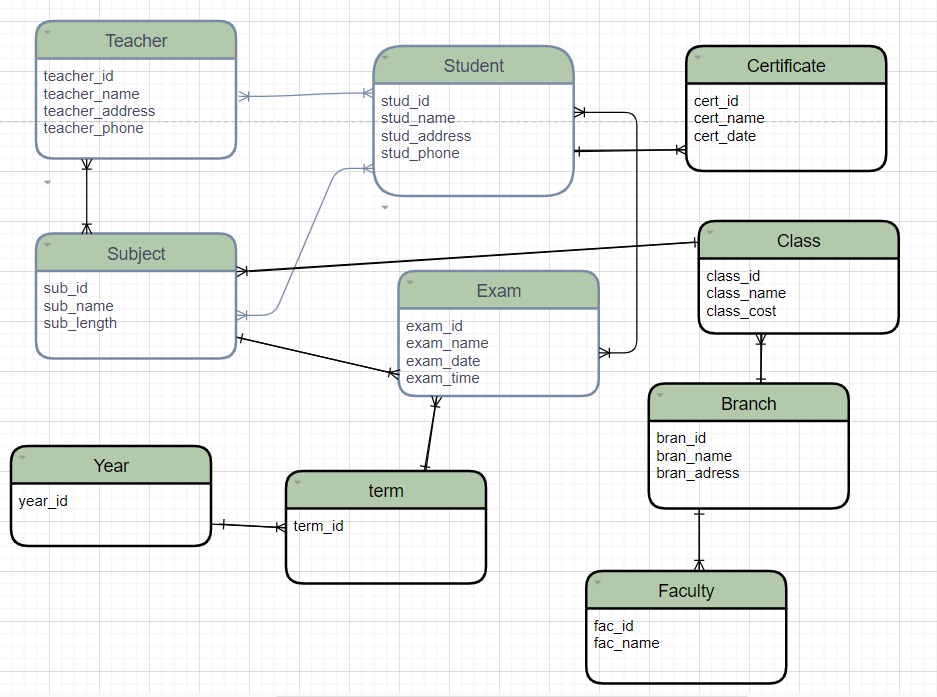
Faculty: fac\_id,fac\_name.

Term: term\_id.

Year: year\_num.

Certificate: cert\_id, cert\_name, cert\_date.

**This is the EAR Diagram:** (by using ERD.drawio)



**The Relationships between Entities are:**

- Has 5 branches.

- There are 4-6 classes in each branch.

- There are 30 students in each class, each student studying one subject at least and takes one or more exams and gets a mark.

- Each class has almost 7-8 subjects.

- Each teacher teaches one or more subjects and each subject teaches by one or more than one teacher.

- There are 4 terms in a year and two tests are conducted in each term.

* **Logical Design:** is an abstract conceptual design. You are not dealing with the physical implementation details; you are merely dealing with identifying the types of information that you need. This logical design process involves arranging all the data into a series of logical relationships called entities andattributes.

- I checked all fields names, and there is no duplication of them.

### - we have to choose the primary key and the foreign key which produces from one-to-many relation and from many to many.

Student

|  |  |  |  |
| --- | --- | --- | --- |
| **Stud\_id** | Stud\_name | Stud\_address | Stud\_phone |

Teacher

|  |  |  |  |
| --- | --- | --- | --- |
| **Teacher\_id** | Teacher\_name | Teacher\_address | Teacher\_phone |

Stud\_teacher

|  |  |  |
| --- | --- | --- |
| Stud\_id  ---------- | Teacher\_id  --------------- | Session\_time |

Subject

|  |  |  |  |
| --- | --- | --- | --- |
| **Sub\_id** | Sub\_name | Sub\_length | Class\_id  ----------- |

Stud\_sub

|  |  |  |
| --- | --- | --- |
| Stud\_id  ----------- | Sub\_id  ---------- | Status |

Teacher\_sub

|  |  |  |
| --- | --- | --- |
| Teacher\_id  --------------- | Sub\_id  --------- | Sub\_date |

Certificate

|  |  |  |  |
| --- | --- | --- | --- |
| **Cert\_id** | Cert\_name | Cert\_date | Stud\_id  ----------- |

Branch

|  |  |  |
| --- | --- | --- |
| **Bran\_id** | Bran\_name | Bran\_address |

Class

|  |  |  |
| --- | --- | --- |
| **Class\_id** | Class\_name | Class\_cost |

Exam

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exam\_id** | Exam\_name | Exam\_date | Exam\_time | Sub\_id  --------- | Term\_id  ----------- |

Stud\_exam

|  |  |  |
| --- | --- | --- |
| Stud\_id  ----------- | Exam\_id  ----------- | mark |

Faculty

|  |  |  |
| --- | --- | --- |
| **Fac\_id** | Fac\_name | Bran\_id  ----------- |

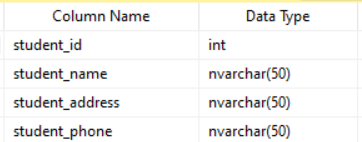
Term

|  |  |
| --- | --- |
| **Term\_id** | Year\_id  ---------- |

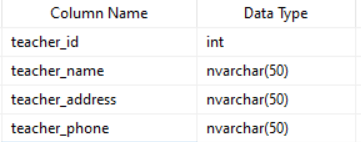
Year

**Year\_id**

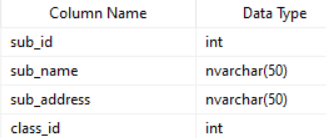
\* We have **Data validation** for each table that is:

Student:  


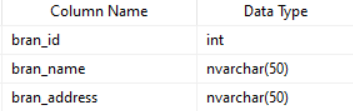
Teacher:



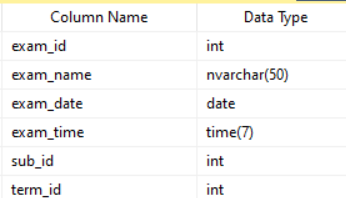
Subject:



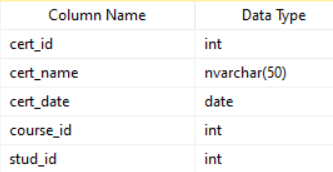
Branch:



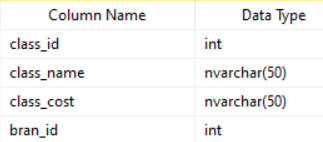
Exam:



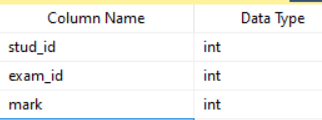
Certificate:



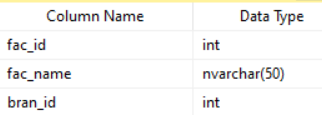
Class:



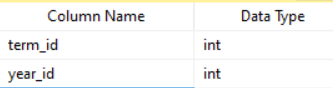
Stud\_exam:



Faculty:



Term:



We had created all tables in this Data Base:

Create table student:

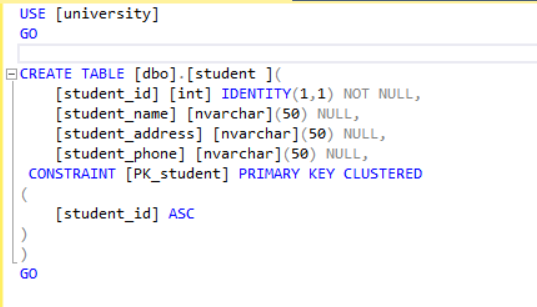
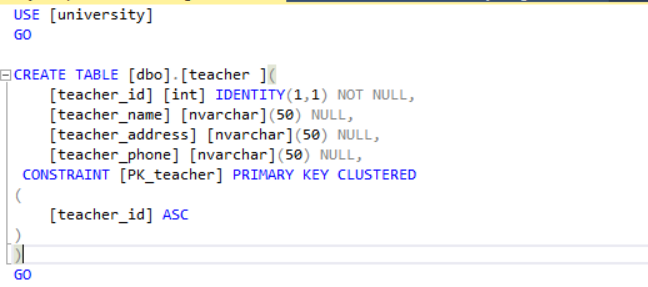
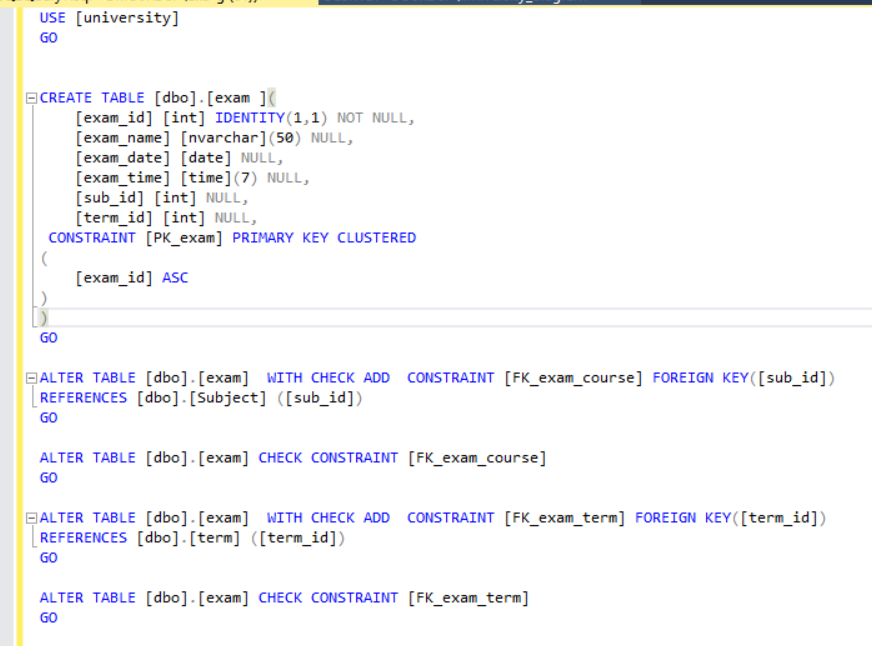


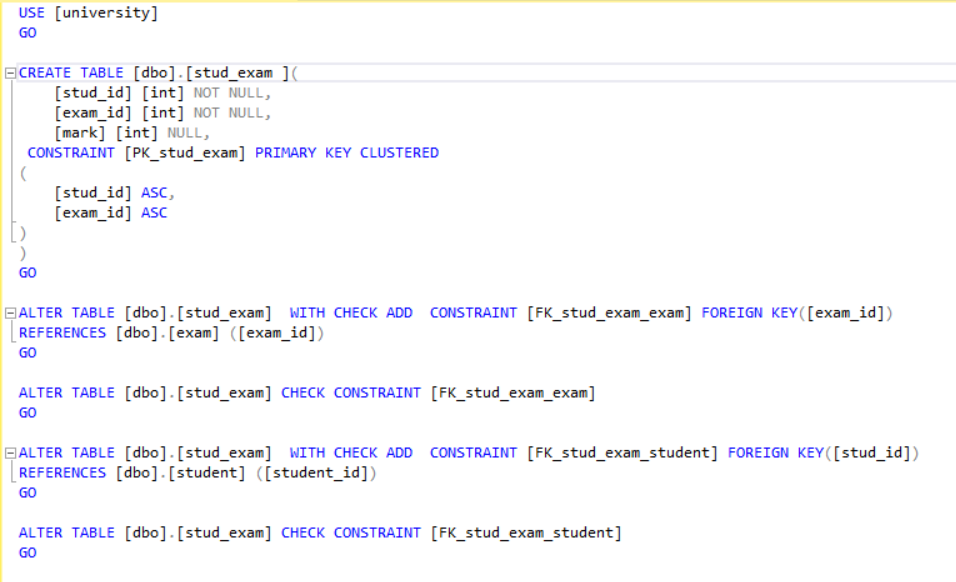
Table teacher:



Exam table:



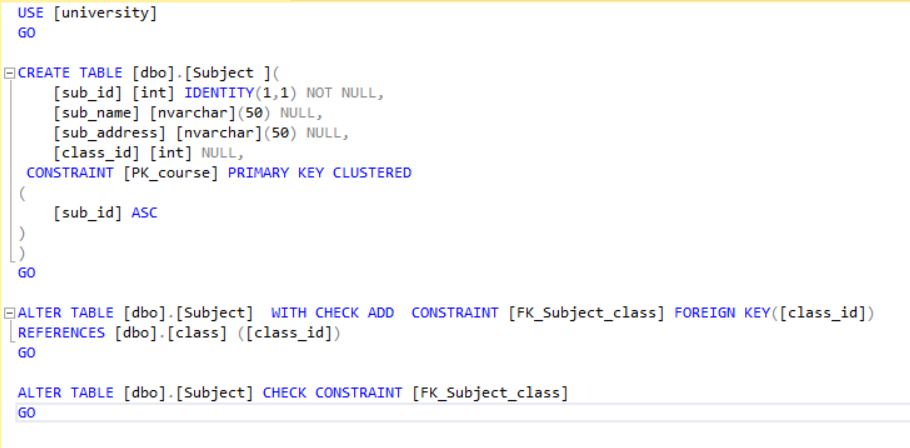
Stud\_exam table:



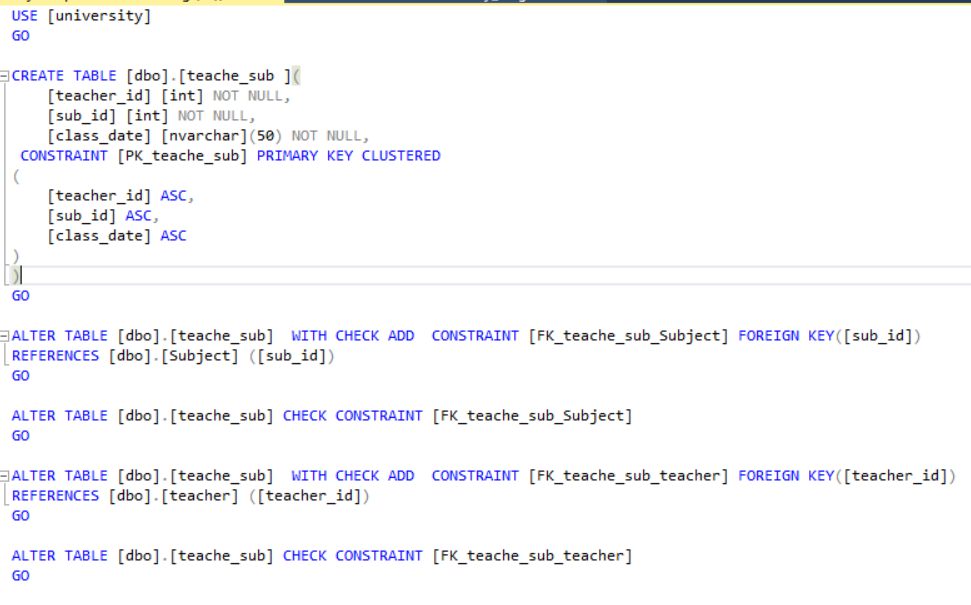
Stud\_teacher table:



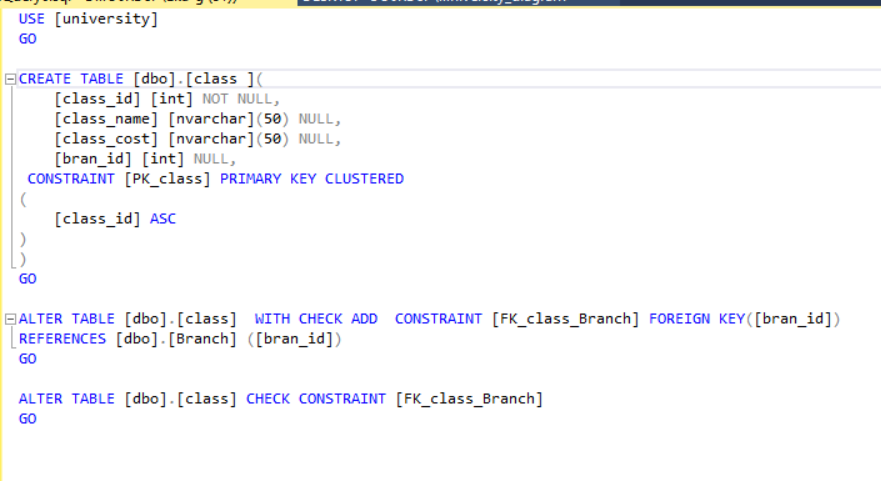
Subject table:



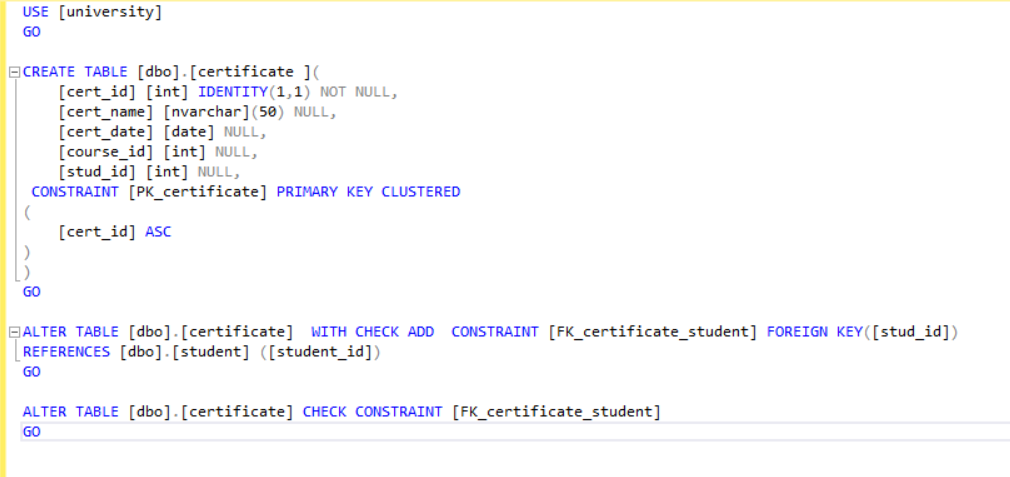
Teacher\_sub table:



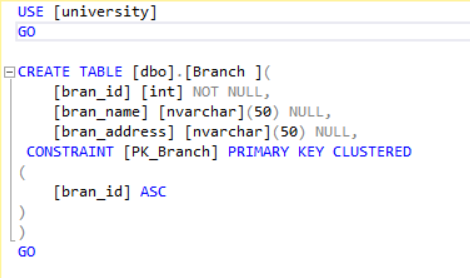
Class table:



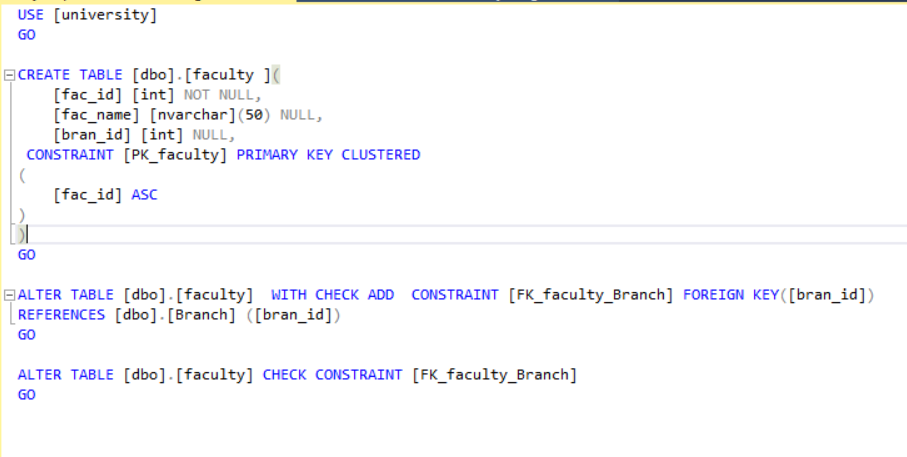
Certificate table:



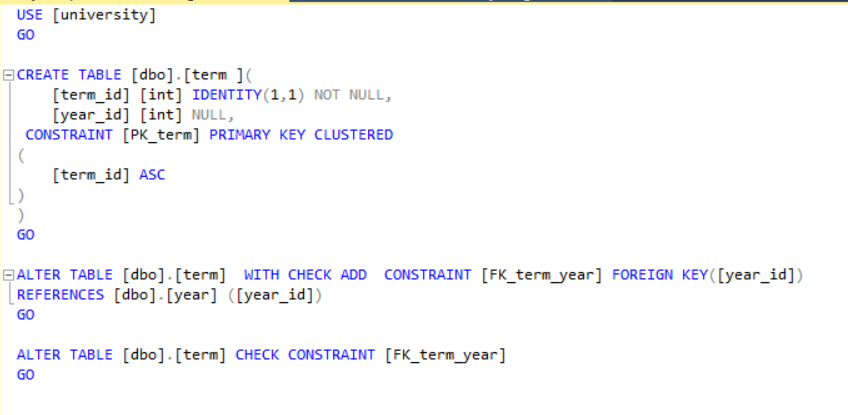
Branch table:



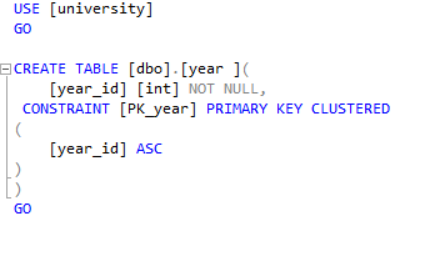
Faculty table:



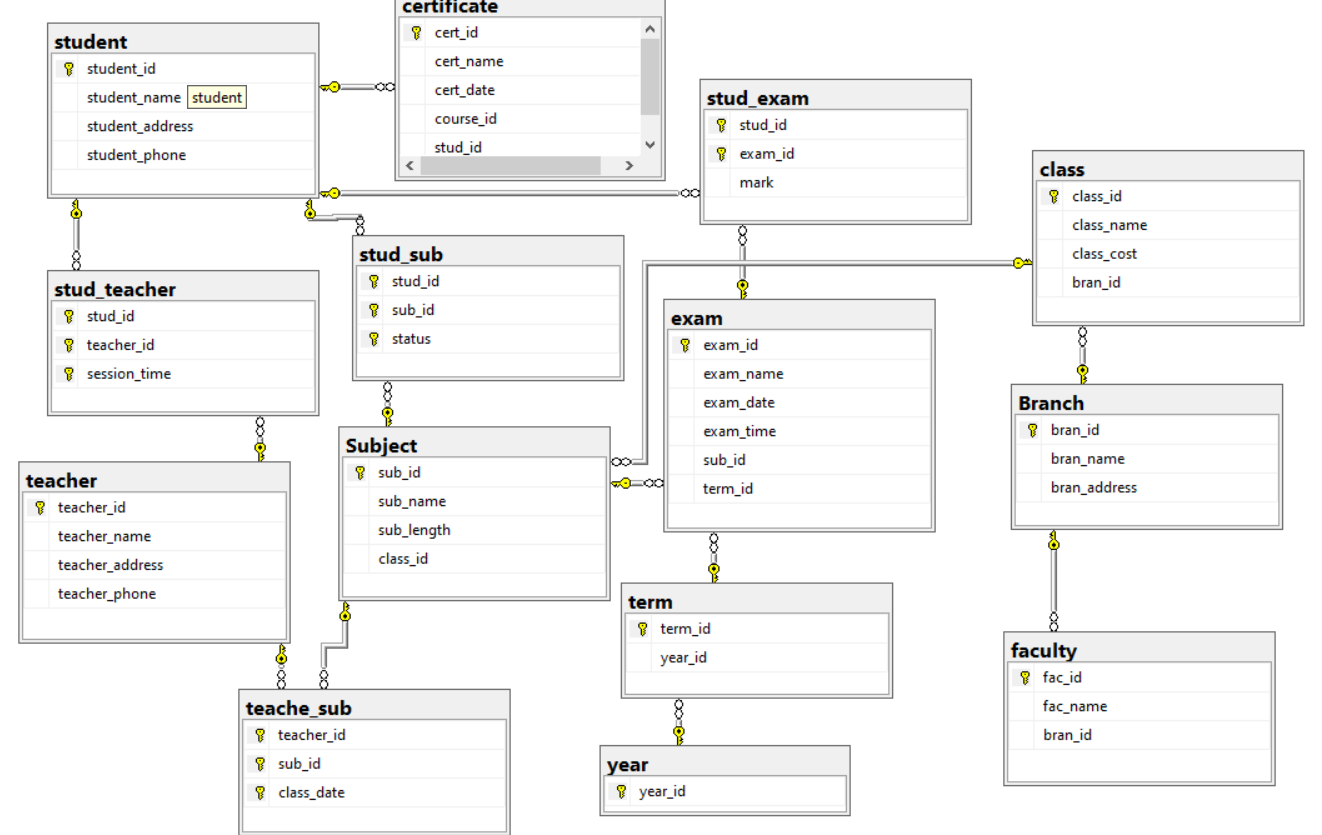
Term table:



Year table:

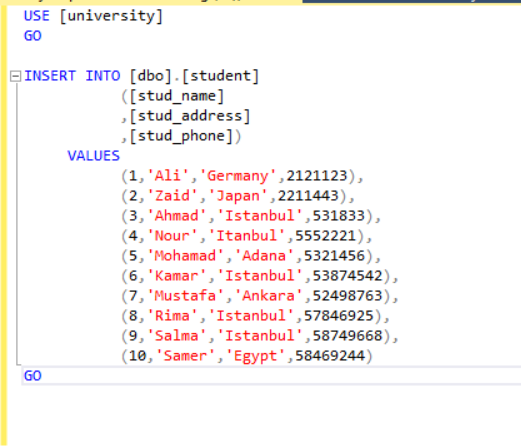


After create all tables the Database Diagram on sql server become:

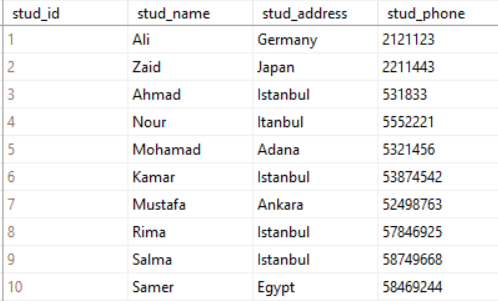


- we have to insert data into the tables by using insert code.

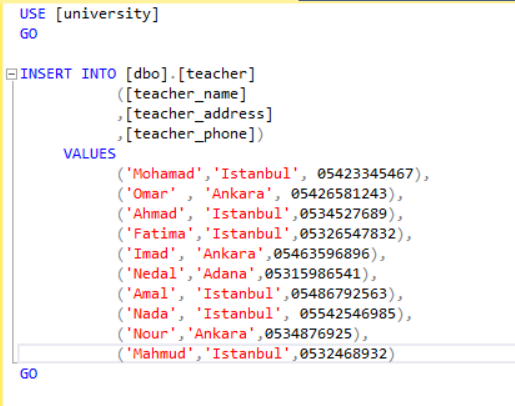
Insert data into student's table:



And the result will be:



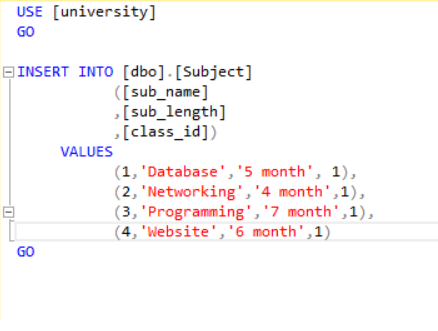
Insert data into teacher's table:



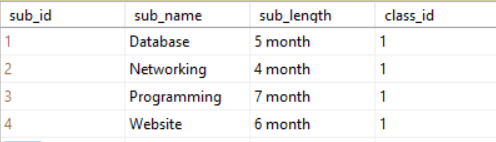
And the result will be:



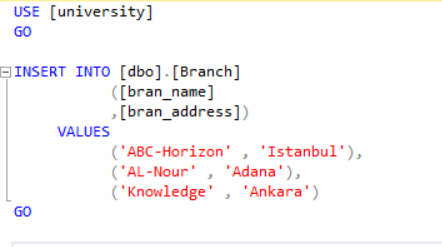
Insert data into Subject's table:



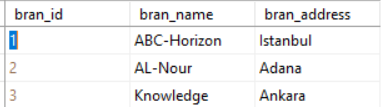
And the result will be:



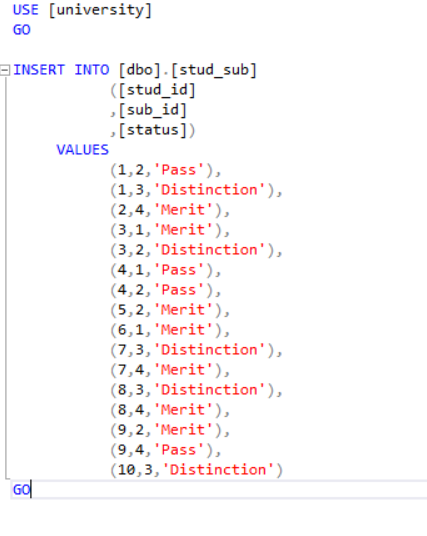
Insert data into Branch's table:



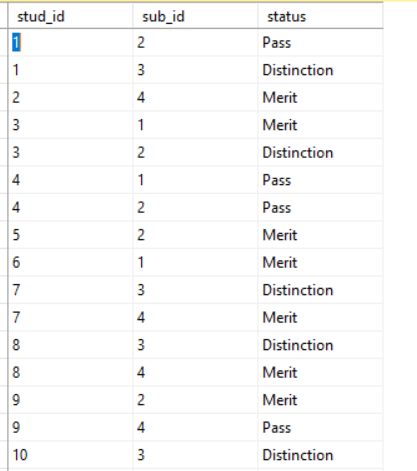
And the result will be:



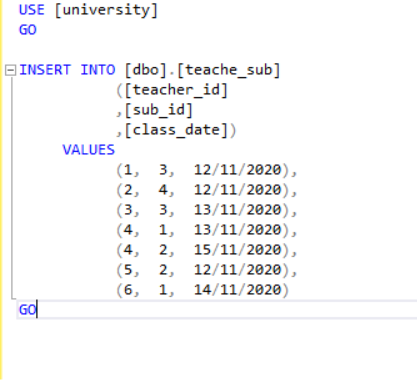
Insert data into stud\_sub's table:



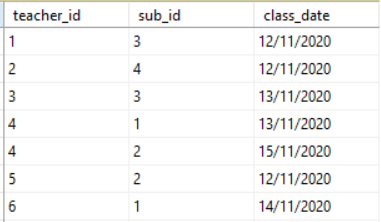
And the result will be:



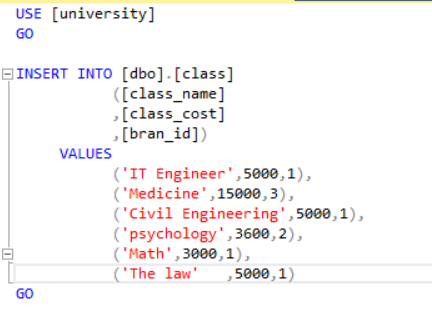
Insert data into teacher\_sub's table:



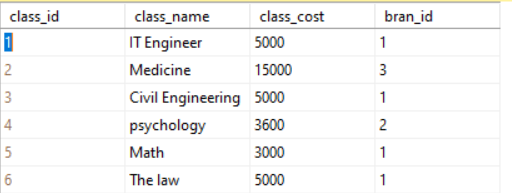
And the result will be:



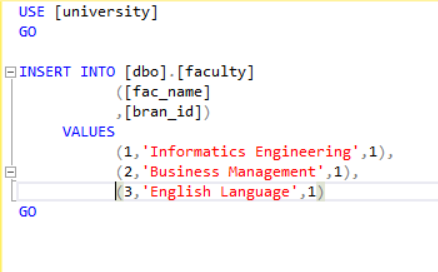
Insert data into class's table:



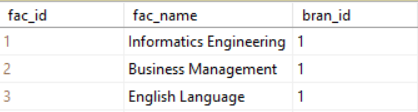
And the result will be:



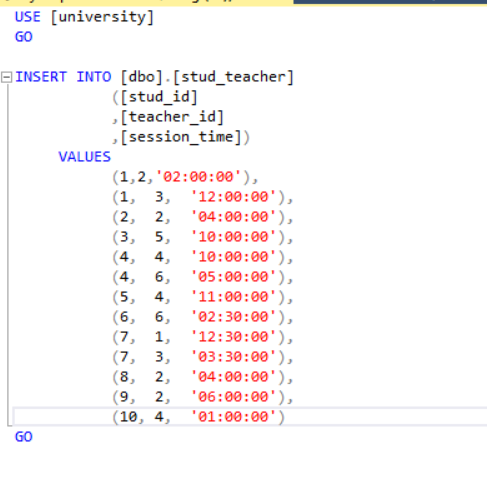
Insert data into faculty's table:



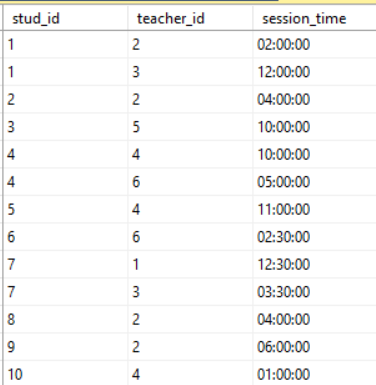
And the result will be:



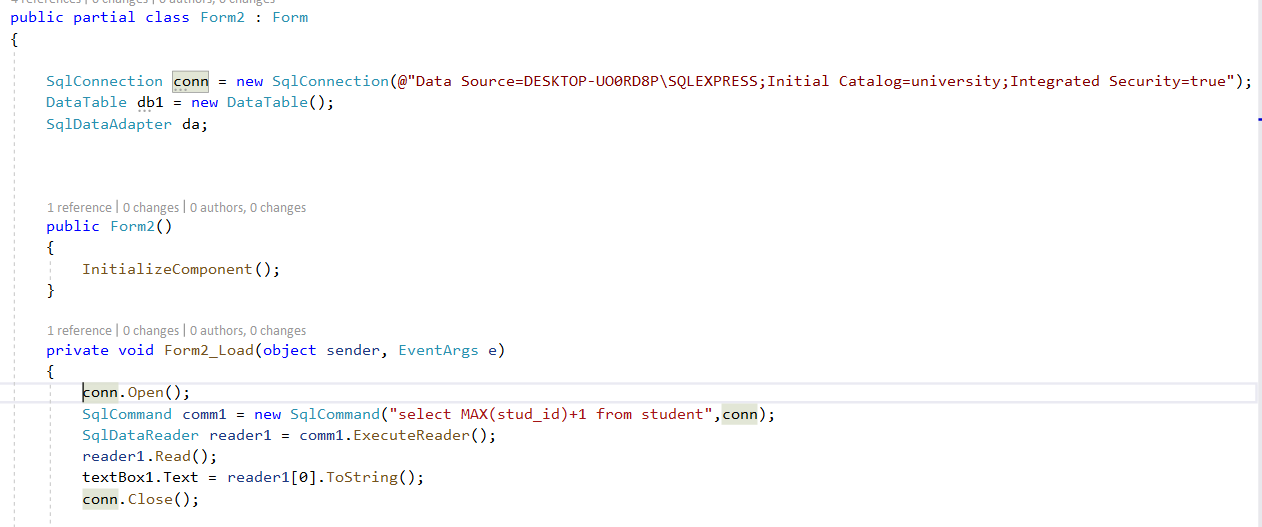
Insert data into stud\_teacher's table:

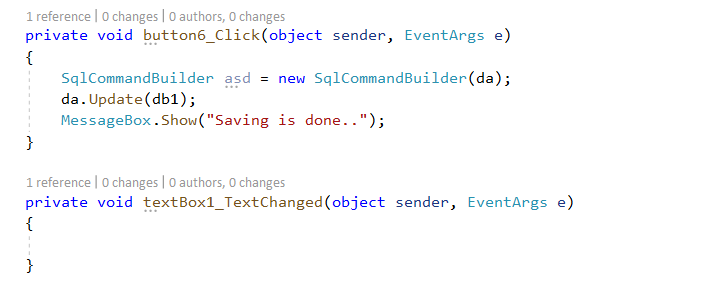
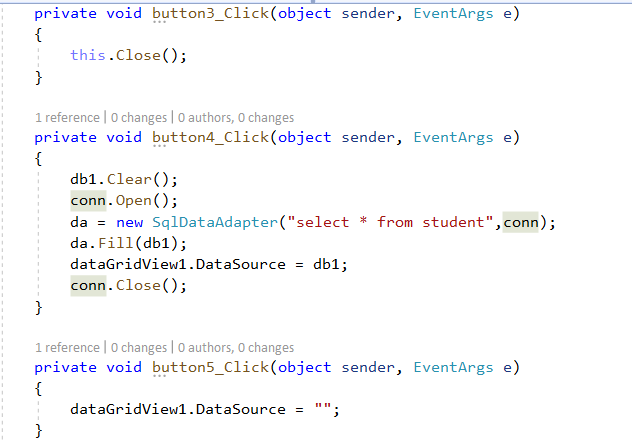


And the result will be:

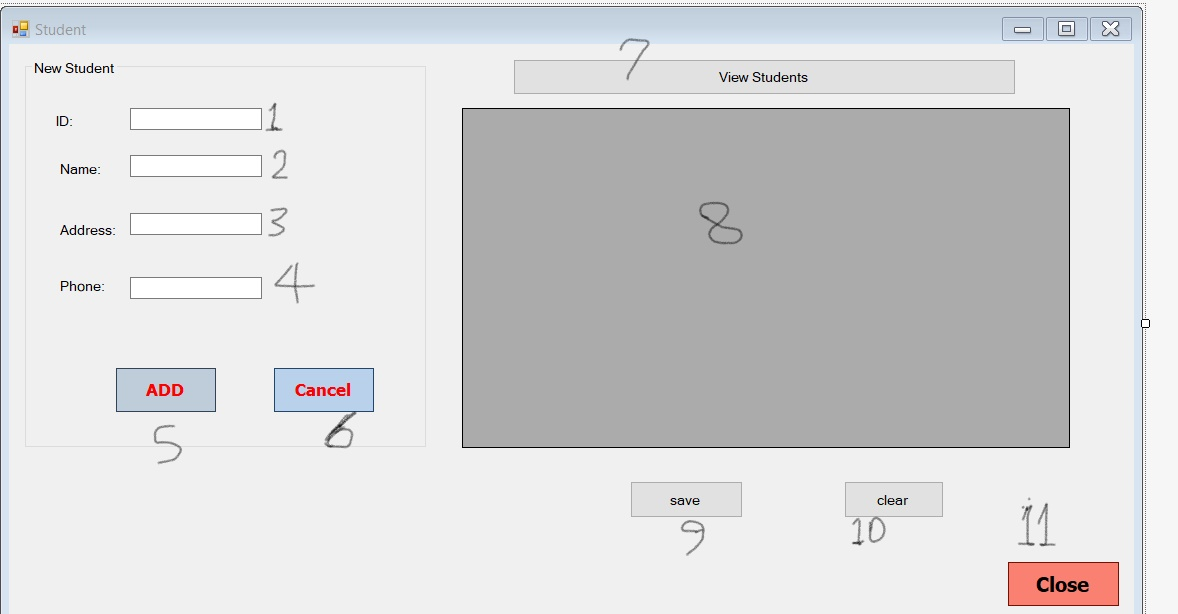


**Student Interface code:**





**User documentation:**



[1] to add student id

[2] to add student name

[3] to add student address

[4]to add student phone

[5] to add student to database

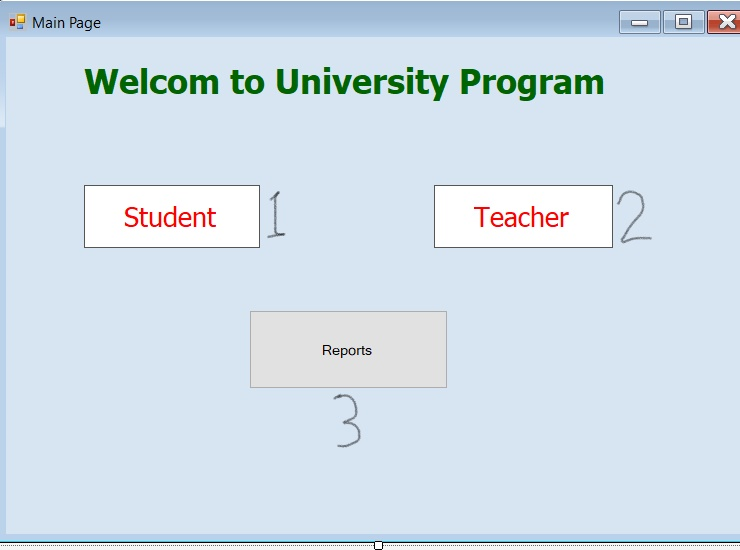
[6] to cancel adding student

[7] to show students in [8]

[9] to save edit or delete student

[10] to clear [8]

[11] to close student form



[1] to get student form

[2] to get teacher form

[3] to get report form

**References:**

|  |
| --- |
| https://docs.oracle.com |