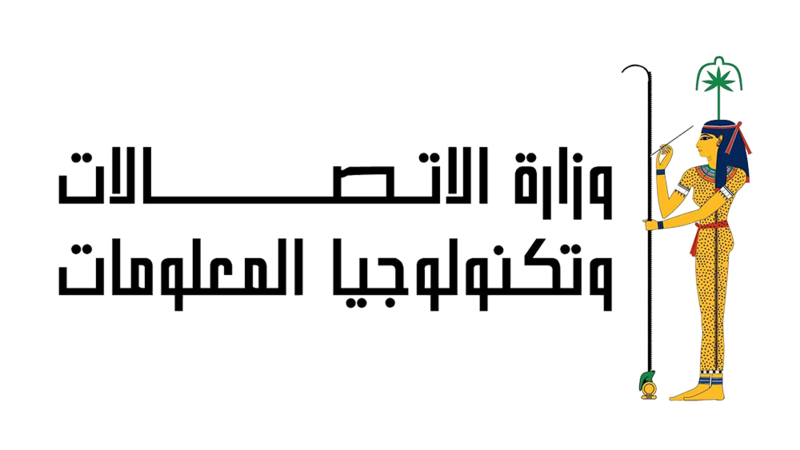
​

*Information Technology Institute*

*.Net Full Stack Developer*

**Examination System**

Supervisor:

Eng. **Sarah Salah**

**Presented by:**

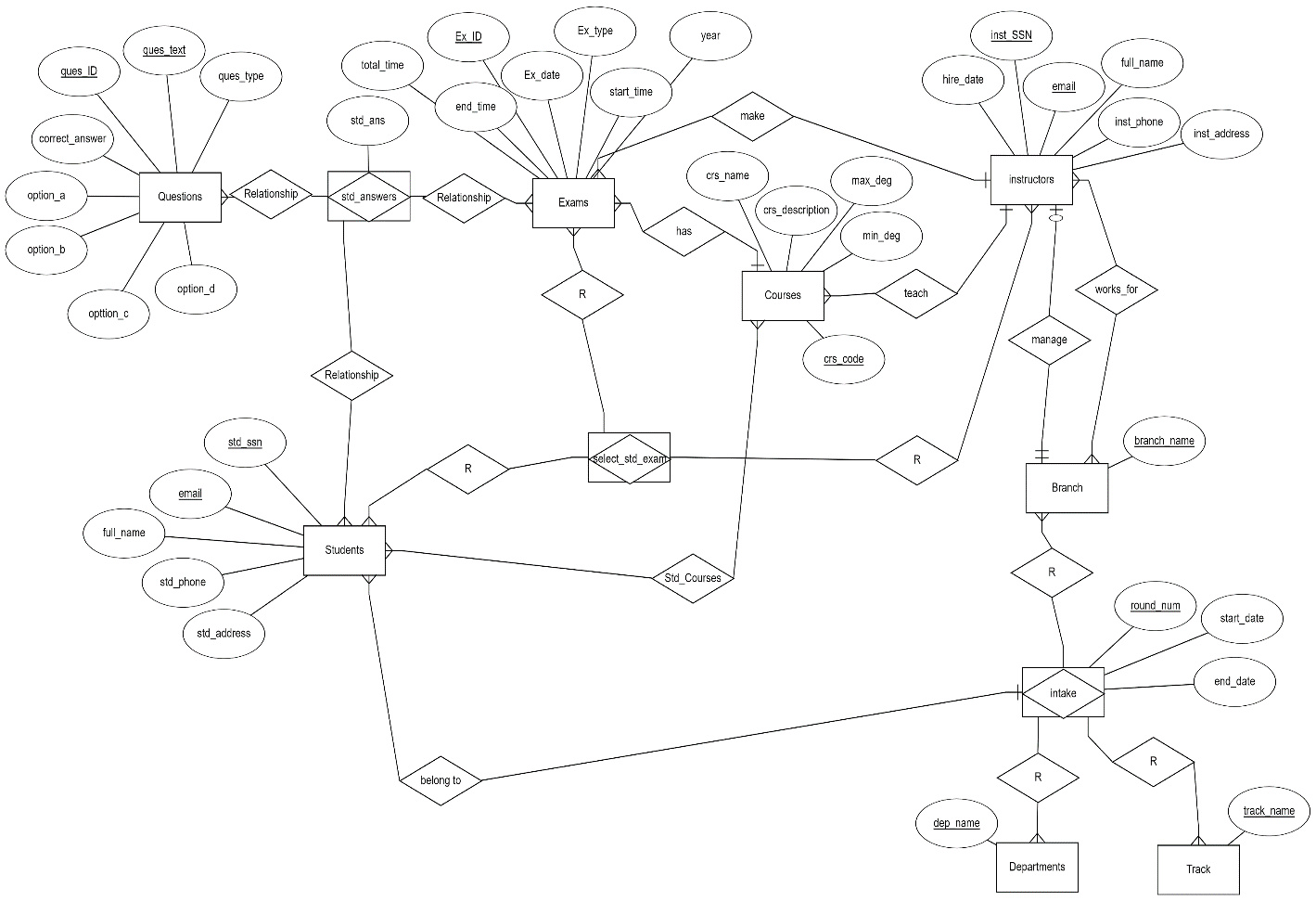
Ahmed Mohamed Abdelwahab

Ahmed Refaat Mohamed

Mohamed Ali Abdelhafez

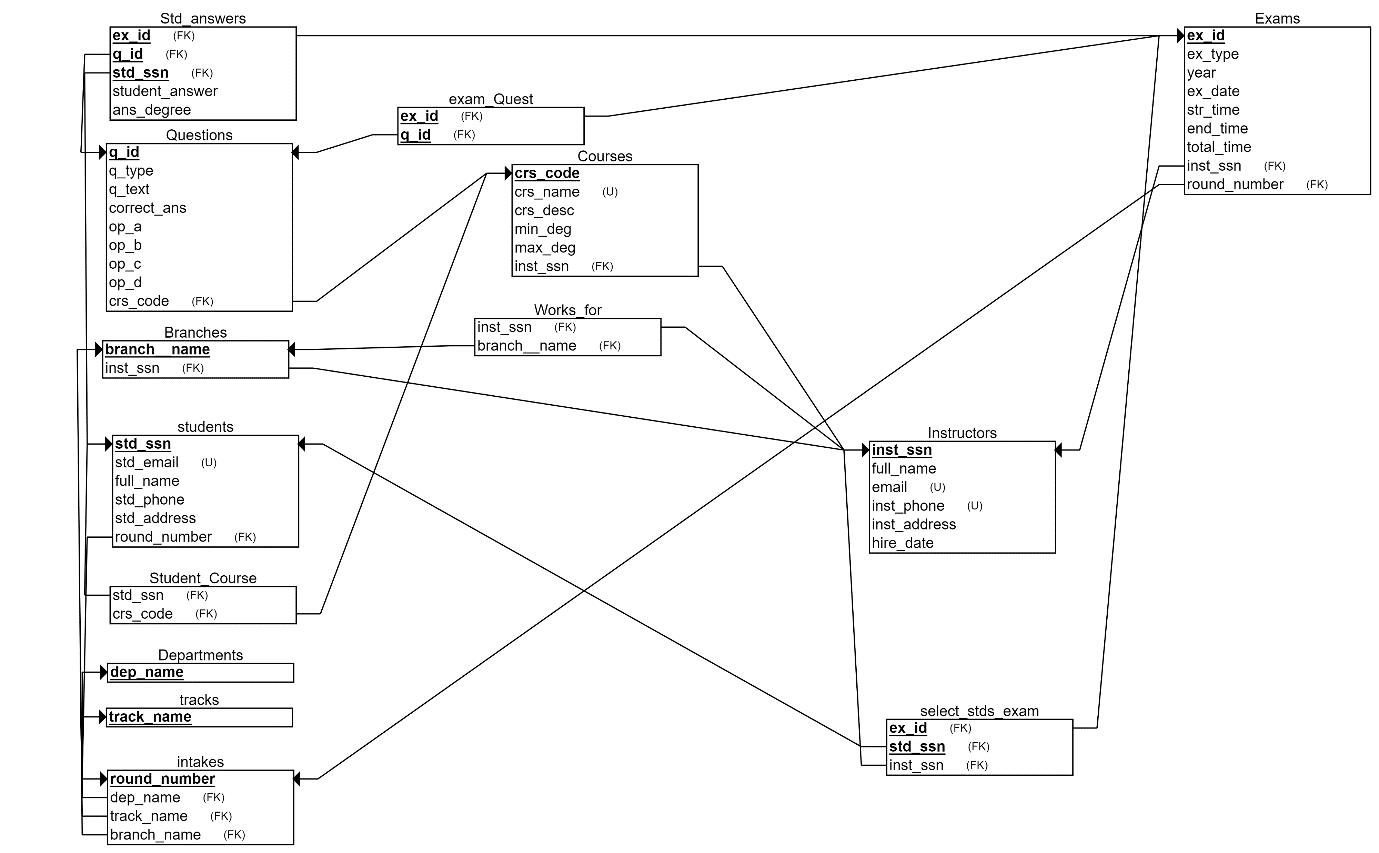
DATABASE DESIGN (ERD)

Examination system ERD includes entities such as Student ,Instructor, Intake, department, students, questions, courses, grades and their relationships to each other.



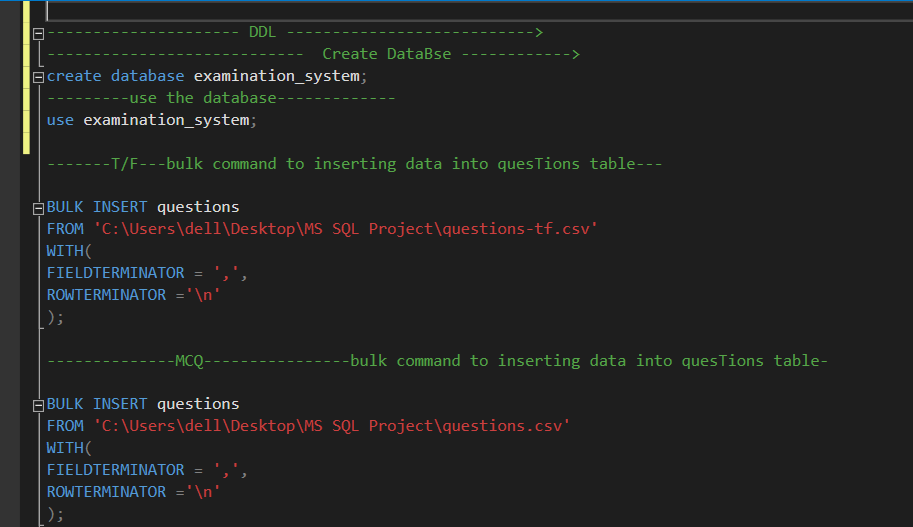
DATABASE DESIGN (MAPPING)

Mapping in the examination system refers to the process of creating a correspondence or association between different entities or attributes in the system needed to implement in the SQL Server Management Studio.



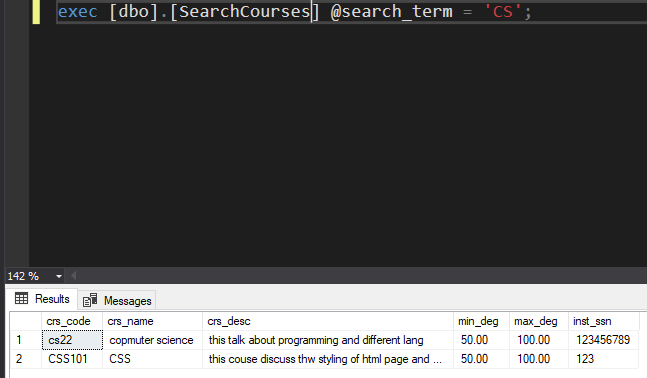
IMPLEMENTATION

Creating a database for an examination system typically involves defining tables to store information about the Topic, Course, Instructor, Student, Department, Exam Once we defined the table schema, we populated the tables with data. We used the Bulk Insert command. This command allows for the efficient insertion of large amounts of data into tables from a data file. The data file can be CSV in our system.

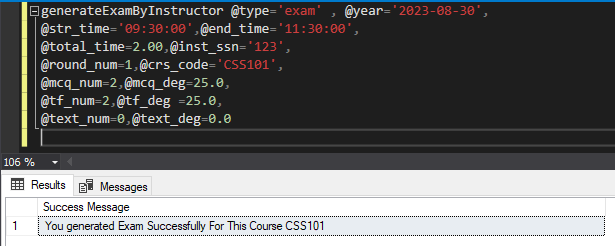


Data Snapshots

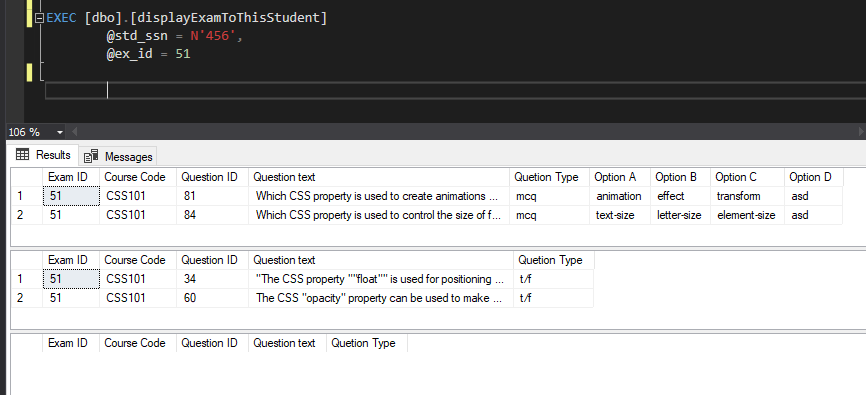
* Search About specific Cousre



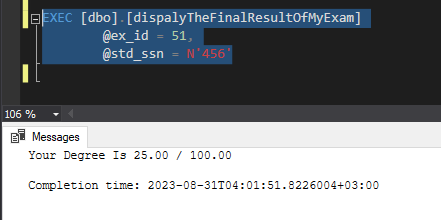
* Generate Exam by selecting rondom questions



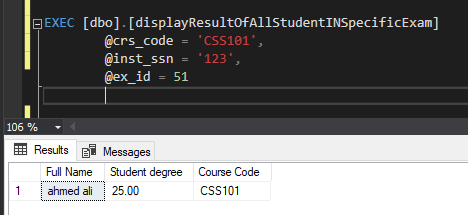
* Student display exam question in its specific time



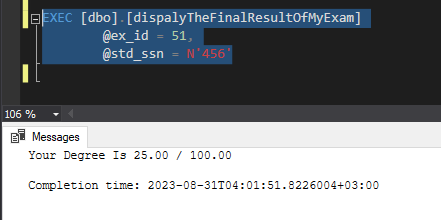
* Student Display his final result exam



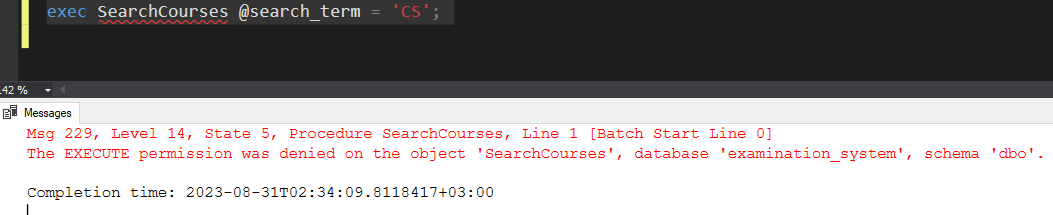
* Instructor display result of all students in specicf exam in his course.



* Display student result in specific exam



* Example to check that permsions is working



Instructor Procedures

* generateExamByInstructor

@type char(10), ----type Exam (exam , corrective)

@year date, ----------------(Date of an exam)

@str\_time time(7), ----------Start time--------

@end\_time time(7), ----------End time--------

@total\_time numeric(2,1), --------total time of Exam ---------

@inst\_ssn char(14), -----------Instructor SSN ---------

@round\_num int, ------------intake round number -----

@crs\_code varchar(25), ----------Course Code ---------

@mcq\_num int, @mcq\_deg numeric(2,1), ---Number of MCQ quesstions ----

@tf\_num int, @tf\_deg numeric(2,1), --------Number of t/f quesstions ----

@text\_num int , @text\_deg numeric(2,1) --Number of text quesstions

Instructors Generates an Exam with Rondom question From questions pool.

* CreateExamWithBasicInfo

@type char(10), ----type Exam

@year date, ----------------(Date of an exam)

@str\_time time(7), ----------Start time--------

@end\_time time(7), ----------End time--------

@total\_time numeric(2,1), --------total time of Exam -----

@inst\_ssn char(14), -----------Instructor SSN ---------

@round\_num int, ------intake round number -----

@crs\_code varchar(25) ------Course Code ----

Instructor Create Exam with The Basics Info.

* displayAllExamForCourseThatInstructorTeach

@inst\_ssn char(14)

Instructor Proc to display all exam for his course teach

* displayAllQuestionsForCourseThatInstructorTeach

@inst\_ssn char(14)

Instructor Proc to display all Questions for his course teach.

* SelectingQuestionToOneByOneExamManually

@ex\_id int , ---Exam ID ---

@q\_id int , ------Question ID ----

@q\_degree numeric(2,1) -----Question Degree -----

Instructor Select the Exam and its Questions one by one.

* displayAllStudentThatEnrrolledInSpecCourse

@crs\_code varchar(25)

instructor display all students that Enrolled in his course that he teach.

* displayResultOfAllStudentINSpecificExam

@crs\_code varchar(25),

@inst\_ssn char(14),

@ex\_id int

Instructor Display Result of All Student In his Course.

Student Procedures

* displayExamToThisStudent

@std\_ssn char(14),

@ex\_id int

This procedure allow student to do and see his Exam in its specific time.

* answerQuestionToSpecificExam

@ex\_id int ,

@q\_id int,

@std\_ssn CHAR(14),

@std\_answer varchar(100)

This procedure allow student to answer exam’s Question in exam time.

* updateAnswerQuestionToSpecificExam

@ex\_id int ,

@q\_id int,

@std\_ssn CHAR(14),

@new\_std\_answer varchar(100)

This procedure allow student to update his answer exam’s Question in exam time.

* dispalyTheFinalResultOfMyExam

@ex\_id int,

@std\_ssn char(14)

This procedure allow student to display his result

In this exam after the exam is expired.

Manager Procedures

* PROCEDURE AddCourse

@crs\_code VARCHAR(25),

@crs\_name NVARCHAR(50),

@crs\_desc VARCHAR(MAX),

@min\_deg numeric(18, 0),

@max\_deg numeric(18, 0),

@inst\_ssn CHAR(14)

It allows the training manager to add new courses through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* PROCEDURE DeleteCourse

@crs\_code VARCHAR(25)

It allows the training manager to Delete course through the following parameter and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* PROCEDURE UpdateCourse

@crs\_code VARCHAR(25),

@NewCourseName VARCHAR(50),

@NewCourseDesc VARCHAR(MAX),

@NewMinDeg numeric(18, 0),

@NewMaxDeg numeric(18, 0),

@NewInstSsn CHAR(14)

It allows the training manager to Upate course data through the following parameter and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* PROCEDURE SearchCourses

@search\_term NVARCHAR(100)

It allows to Search about course through the following parameter and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* PROCEDURE SearchStudents

@search\_term NVARCHAR(100)

It allows to Search about Instructor through the following parameter and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* CREATE PROCEDURE Addinstructor

@instssn CHAR(14),

@Name VARCHAR(50),

@Email VARCHAR(50),

@Phone VARCHAR(11),

@Adress VARCHAR(100),

@Hiredate DATE  
  
It allows the training manager to add new Instructors through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* CREATE PROCEDURE UpdateInstructor

@instssn CHAR(14),

@NewName VARCHAR(50),

@NewEmail VARCHAR(50),

@NewPhone VARCHAR(11),

@NewAddress VARCHAR(100),

@NewHiredate DATE

It allows the training manager to Update in Instructors Data through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* CREATE PROCEDURE DeleteInstructor @instssn CHAR(14)

It allows the training manager to Delete Instructor through the following parameter and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

## *this procedure can create a New intakes*

alter procedure add\_new\_intake

@round\_num int , @str\_date date, @end\_date date , @dep\_name varchar(100) , @track\_name varchar(100) ,@branche\_name varchar(100)

It allows the training manager to add new intakes through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

### *this procedure can delete exist intake*

alter procedure delete\_intake (@round\_num int )

It allows the training manager to delete intakes through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedureupdate\_intake\_info

(@round\_num int ,

@str\_date date,

@end\_date date ,

@dep\_name varchar(100) ,

@track\_name varchar(100) ,

@branch\_name varchar(100))

It allows the training manager to updating intakes information through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure add\_new\_student

@std\_ssn char(14) ,

@str\_email varchar(50),

@full\_name varchar(50) ,

@std\_phone char(11) ,

@std\_address varchar(100) ,

@round\_num int)

It allows the training manager to add new students through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure delete\_student

@std\_ssn char(14)

It allows the training manager to delete students through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure update\_student\_info

(@std\_ssn char(14) ,

@std\_email varchar(50),

@full\_name varchar(50) ,

@std\_phone char(11) ,

@std\_address varchar(100) ,

@round\_num int)

It allows the training manager to updating students informations through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure ADD\_BRANCHES

(@branch\_name as varchar(100),

@inst\_ssn as char(14))

It allows the training manager to add new branches through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure delete\_branch

@branch\_name varchar(100)

It allows the training manager to delete branches through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure update\_branch\_info

(@branch\_name varchar(100),

@inst\_ssn char(14) ,

@new\_branch\_name varchar(100) ,

@new\_inst\_ssn char(14))

It allows the training manager to updating branches informations through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure ADD\_TRACKS

(@TRACK\_name as varchar(100))

It allows the training manager to add new trackes through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure delete\_track

@track\_name varchar(100)

It allows the training manager to delete trackes through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure update\_track\_info

(@old\_track\_name as varchar(100),

@new\_track\_name as varchar(100))

It allows the training manager to updating trackes informations through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* PROCEDURE ADD\_DEPARTMENT

(@DEPARTMENT\_NAME as varchar(100))

It allows the training manager to add new Departments through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure delete\_department

@DEPARTMENT\_NAME varchar(100)

It allows the training manager to delete departments through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure update\_department\_info

(@old\_department\_name varchar(100) ,

@new\_department\_name varchar(100))

It allows the training manager to updating departments informations through the following parameters and by checking and ensuring that the data is correct and does not conflict with the requirements and conditions of the system.

* procedure add\_question\_pool

(@inst\_ssn char(14),

@q\_id int ,

@q\_text varchar(max),

@crs\_code varchar(25),

@q\_type varchar(5),

@correct\_ans varchar(100),

@op\_a varchar(100) ,

@op\_b varchar(100),

@op\_c varchar(100) ,

@op\_d varchar(100))

This allows the instructor to add questions related to his course only and does not allow him to add any questions in the rest of the courses

* PROCEDURE Ubdate\_Question

@inst\_ssn char(14),

@q\_id int ,

@q\_text varchar(max),

@crs\_code varchar(25),

@q\_type varchar(5),

@correct\_ans varchar(100),

@op\_a varchar(100),

@op\_b varchar(100),

@op\_c varchar(100),

@op\_d varchar(100))

This allows the instructor to updating questions related to his course only and does not allow him to updating any questions in the rest of the courses.

* procedure delete\_question

@inst\_ssn char(14), @q\_id int ,

@crs\_code varchar (25)

This allows the instructor to delete questions related to his course only and does not allow him to delete any questions from the rest of the courses.

Manager Views

* displayAllInstructors

View To allow manager to Display all Instructor.

* displayAllStudents

View To allow manager to Display all students.