**Examination system**

Group 7

The system has the following tables:

* Branch
* Intake
* Track
* Student
* Instructor
* Course
* studentCourse
* Exam
* Question
* questionExam
* examStudent

**procedures for branch, intake, track (training manager)**

display procedures

* disp\_branch\_proc: displays all the branches
* disp\_track\_proc: displays all the tracks
* disp\_intake\_proc: displays all the intakes
* disp\_all\_proc: displays the branches with the tracks and intakes they have. All with name and id

inserting procedures

* add\_branch\_proc: adds a branch. Takes the name of the branch as an argument of type nvarchar(max)
* add\_intake\_proc: adds an intake. Takes the name of the intake as an argument of type nvarchar(max)
* add\_track\_branch\_proc: adds a track in intake & branch. Takes the name of the track, intake, branch as an argument. All the arguments are of type nvarchar(max)

updating procedures

* edit\_branch\_name\_proc: changes branch name. Takes the oldname, and newname both are of type nvarchar(max)
* edit\_intake\_name\_proc: changes intake name. Takes oldname, newname and branchname all are of type nvarchar(max)
* edit\_track\_name\_proc: changes track name. Takes oldname, newname, intakename and branchname all are of type nvarchar(max)

**procedures for students (training manager)**

display procedures

* disp\_student\_proc: displays all the students

inserting procedures

* add\_student\_proc: adds a student. Takes @name of type nvarchar(max), @uni of type nvarchar(max), @phone of type char(11), @gradDate of type date,

@track\_name of type nvarchar(max), @intake\_name of type nvarchar(max), @branch\_name of type nvarchar(max)

updating procedures

* delete\_student\_proc: deletes a student. Takes @name of type nvarchar(max),

@track\_name of type nvarchar(max), @intake\_name of type nvarchar(max), @branch\_name of type nvarchar(max)

**procedures for instructor (training manager)**

display procedure

* disp\_instructor\_proc: displays all instructors

inserting procedure

* add\_instructor\_proc: adds an instructor. Takes @name of type nvarchar(max)

updating procedure

* edit\_instructor\_proc: updates name of an instructor. Takes @old\_name of type nvarchar(max), @new\_name of type nvarchar(max), @id of type int

delete procedure

delete\_instructor\_proc: deletes an instructor, prints ‘Instructor with ins\_id does not exist’ if the instructor isn’t saved. Takes @ins\_id of type int, @name of type nvarchar(max)

**procedures for courses (training manager)**

display procedures/functions

* disp\_course\_proc: displays courses
* get\_instructor\_courses\_name\_id\_func: gets instructor name, courses they teach, desc of course, mindegree, maxdegree. Takes @ins\_id of type int, @name of type nvarchar(max)
* get\_course\_info\_by\_name\_fun: gets course info by the name of the course. Takes @course\_name of type nvarchar(max)

inserting procedures

* add\_course\_proc: adds a course. Takes @course\_name nvarchar(max),

@description nvarchar(max), @min\_degree int, @max\_degree int,

@instructor\_id int

Updating procedures

* edit\_course\_proc: edits course. Takes @course\_name nvarchar(max),

@description nvarchar(max),@min\_degree int, @max\_degree int,

@instructor\_id int

Deleting procedures

* delete\_course\_proc: deletes course. Takes @course\_id int

**views for courses for tracks(for the 2023Q2 intake in Minya only)**

* courses\_for\_BI: has the courses for the BI track. Courses with id (1,2,3,4,6)
* courses\_for\_python: has the courses for the python track. Courses with id (1,2,4)
* courses\_for\_dot\_net: has the courses for the .net track. Courses with id (1,7)
* courses\_for\_web: has the courses for the web track. Courses with id (7)

**procedures/functuons for courses for tracks(for the 2023Q2 intake in Minya only)**

display function

* view\_student\_courses\_func: displays student names and courses they take

insert procedure

* insert\_course\_student\_BI\_proc: insert courses for student in the BI track
* insert\_course\_student\_py\_proc: insert courses for student in the python track
* insert\_course\_student\_dot\_proc: insert courses for student in the .net track
* insert\_course\_student\_web\_proc: insert courses for student in the web track
* insert\_all\_student: exec all the previous procs

**making exam procedures**

* make\_exam\_proc: makes a new exam. Only allows instructor that teach the course to make it otherwise prints ‘Can only make exam for your course’. After insert prints ‘to chose questions by selecting number of questions of each type and the system selects the questions random exec chose\_Q\_random\_proc'

'to manually chose and view questions exec chose\_Q\_manual\_proc'’

* + Takes @ins\_id int, @c\_id int, @type bit, @year date, @allownace nvarchar(max), @startTime datetime, @endTime datetime
* chose\_Q\_random\_proc: choses question automatically takes number of question in each category and degrees of each category. Takes @exam\_id int, @c\_id int,@NMCQ int, @NTF int, @NTXT int, @MCQ\_degree int, @TF\_degree int, @TXT\_degree int
  + if exam doesn’t exist it prints ‘make exam first’
  + if number of questions chosen is higher than question available it prints ‘chose lower number of questions’
  + if degrees are higher than max degrees of course prints ‘sum of degree must be lower than max degree in course’
  + uses NEWID() for random
* disp\_questions\_proc: displays available questions for manual choosing of question
* make\_exam\_manual\_proc: choses question manually by inserting a string separated by commas for question ids and their degrees. Takes @exam\_id int, @c\_id int, @Question\_id\_string nvarchar(max), @question\_degree\_string nvarchar(max)
  + if exam doesn’t exist it prints ‘make exam first’
  + if ids are higher than the max id available for questions it prints ‘'chose ids from question table'’
  + if degrees are higher than max degrees of course prints ‘'cant make max degrees bigger than max degree for course'’

**assigning exam to students proc**

* select\_students\_for\_exam\_proc: assigns exam to student. Takes @student\_id int, @exam\_id int
  + if student id doesn’t exist prints ‘'Student does not exist'’
  + if exam id does not exist prints 'Exam does not exist'
  + if student is already assigned this exam prints ‘'Student already assigned to this exam'’

**checking access to exam**

* CheckExamAccess\_proc: checks if student has this exam and if this the time for it.
  + Takes @examId int, @studentId int
  + If the student isn’t assigned this exam prints 'You do not have access to this exam'
  + if the student opens the exam in a wrong time it prints ‘The exam is not available at this time'

**store student answers for exam**

* StoreStudentAnswer\_proc: stores student answers for each question id
  + Takes @studentId int, @examId int, @questionId int, @answerText nvarchar(max)
  + If student doesn’t have access to exam print 'You do not have access to this exam'
  + If student enters a question id that isn’t in the exam prints 'The question does not belong to this exam'
  + If all is right prints 'Your answer has been stored'

**Calculate student answer score for exam**

* check\_answer\_\_calc\_score\_proc: calculates if the answer if right
  + takes @exam\_id int, @student\_id int
  + if the answer is wrong then the score is zero otherwise it’s the full score of the question

**Calculate student final score for course**

* calc\_final\_result\_for\_student: calculates the sum of the scores in the exam
  + takes @exam\_id int,@student\_id int

some extra views:

* ins\_course: shows courses that instructors teach
* intake\_track: shows tracks that intakes have
* studentcourses: shows courses that students have