none

Examination System Data Dictionary

1/20/2021





Table of contents

xamina	ation System	8
1. Tab	oles	9
1.1.	Table: Answer (Table of Answers)	9
1.2.	Table: Course (Table of Courses)	9
1.3.	Table: Course_Topics (Course & Topic)	10
1.4.	Table: Department (Table of departments)	11
1.5.	Table: Dept_Course (Department & Course)	12
1.6.	Table: Dept_Stud (Departments & Students)	13
1.7.	Table: Exam (Table of Exams)	14
1.8.	Table: Exam_Ques (Exam & Question)	15
1.9.	Table: Instructor (Table of Instructors)	15
1.10.	. Table: Question (Table of Questions)	16
1.11.	Table: Regis_Inst (Registrar & Instructor)	17
1.12.	. Table: Regis_Stud (Registrar & Student)	18
1.13.	. Table: Registrar (Table of users information (Registeration))	19
1.14.	. Table: St_exam_Q_A (Student & Exam & Question)	20
1.15.	. Table: Stud_Course (Student & Course)	21
1.16.	. Table: Stud_Exam (Student & Exam)	22
1.17.	. Table: Student (Table of Students)	23
1.18.	. Table: Topic (Table of Topics)	24
2. Pro	ocedures	25
2.1.	Procedure: deleteAnswer (Delete an answer)	25
2.2.	Procedure: deleteCourse (Delete a course)	25
2.3.	Procedure: deleteCourseTopic (Delete a topic in a course)	25
2.4.	Procedure: deleteDepartment (Delete a department)	26
2.5.	Procedure: deletedepartmentCourse (Delete a course in a department)	26
2.6.	Procedure: deleteDepartmentStudent (Delete a student in a department)	26
2.7.	Procedure: deleteExam (Delete an exam)	27
2.8.	Procedure: deleteExamQuestion (Delete a question in an exam)	27
2.9.	Procedure: deleteInstructor (Delete an instructor)	27
2.10.	Procedure: deleteQuestion (Delete a question)	28
2.11.	Procedure: deleteRegisterInstructor (delete an instructor user)	28
2.12.	Procedure: deleteRegisterStudent (Delete a student user)	28
2.13.	Procedure: deleteRigstrar (delete user information)	29
2.14.	Procedure: deleteStudent (Delete a student)	29
2.15.	Procedure: deleteStudentExamQuestionGradeAnswer (Delete a row in St_exam_Q_A table)	29
2.16.	Procedure: deleteStudentperCourse (Delete student per course)	30
2.17.	. Procedure: deleteStudentperExam (Delete student per exam)	30
2.18.	Procedure: deleteTopic (Delete a topic)	30
2.19.	Procedure: examAnswer (Student' Answers of the exam)	31
2.20). Procedure: examCorrection (Exam Correction)	31
2.21.	Procedure: examGeneration (Exam Generation)	31
2.22	2. Procedure: getAllAnswers (Retrieve all answers)	32
2.23	3. Procedure: getAllCourses (Retrieve all Courses)	32
2.24	4. Procedure: getAllCoursesTopics (Retrieve all topics of a course)	32

2.25.	Procedure: getAllDepartments (Retrieve all departments)	32
2.26.	Procedure: getAllDepartmentscourse (Retrieve all courses per department)	33
2.27.	Procedure: getAllDepartmentsStudents (Retrieve all students per department)	33
2.28.	Procedure: getAllExams (Retrieve all exams)	33
2.29.	Procedure: getAllExamsQuestions (Retrieve all questions per exam)	33
2.30.	Procedure: getAllInstructors (Retrieve all instructors)	33
2.31.	Procedure: getAllQuestions (Retrieve all questions)	34
2.32.	Procedure: getAllRegisterInstructors (Retrieve all instructors and their the user ids)	34
2.33.	Procedure: getAllRegisterStudents (Retrieve all students and their the user ids)	34
2.34.	Procedure: getAllRegistrars (Retrieve all user's info)	34
2.35.	Procedure: getAllStudents (Retrieve all students)	35
2.36.	Procedure: getAllStudentsExamsQuestionsGradesAnswers (Retrieve all exams' answers for students)	35
2.37.	Procedure: getAllStudentsperCourses (Retrieve all students per courses)	35
2.38.	Procedure: getAllStudentsperExams (Retrieve all students per exam)	35
2.39.	Procedure: getAllTopics (Retreive all Topics)	36
2.40.	Procedure: insertAnswer (Insert an answer)	36
2.41.	Procedure: insertCourse (Insert a course)	36
2.42.	Procedure: insertCourseTopic (Insert a topic in a course)	36
2.43.	Procedure: insertDepartment (Insert a department)	37
	Procedure: insertDepartmentCourse (Insert a course in a department)	
2.45.		
2.46.		
2.47.	Procedure: insertExamQuestion (Insert a question in an exam)	38
2.48.		
2.49.	Procedure: insertQuestion (Insert a question)	39
2.50.	Procedure: insertRegisterInstructor (Insert an Instructor user)	
	Procedure: insertRegisterStudent (Insert a student user)	
	Procedure: insertRegistrar (Insert a user's info)	
	Procedure: insertStudent (Insert a student)	
2.54.	Procedure: insertStudentExamQuestionGradeAnswer (Insert a student's answer in an Exam's Question)	41
2.55.	Procedure: insertStudentperCourse (Insert a student per course)	42
2.56.	Procedure: insertStudentperExam (Insert Student per Exam)	42
2.57.	Procedure: insertTopic (Insert topic)	42
2.58.	Procedure: NCourse_NumStud	43
2.59.	Procedure: questions	43
2.60.	Procedure: questions_studAnswer	43
2.61.	Procedure: stud_grade	43
2.62.	Procedure: stud_info	44
	Procedure: topics	
2.64.	Procedure: updateAnswer (Update Answer)	44
	Procedure: updateCourse (Update Course)	
	Procedure: updateCourseTopic (Update Course Topic)	
	Procedure: updateDepartment (Update Department)	
	Procedure: updateDepartmentCourse (Update Department Course)	
	Procedure: updateDepartmentStudent (Update Department Student)	
	Procedure: updateExam (Update Exam)	

2.71.	Procedure: updateExamQuestion (Update Exam Question)	47
2.72.	Procedure: updateInstructor (Update Instructor)	47
2.73.	Procedure: updateQuestion (Update Question)	. 48
2.74.	Procedure: updateRegisterInstructor (Update Registered Instructor)	. 48
2.75.	Procedure: updateRegisterStudent (Update Registered Student)	. 48
2.76.	Procedure: updateRegistrar (Update Registration)	49
2.77.	Procedure: updateStudent (Update Student)	. 49
2.78.	Procedure: updateStudentExamQuestionGradeAnswer (Update Student Exam Question Grade Answer)	. 49
2.79.	Procedure: updateStudentperCourse (Update Student per Course)	50
2.80.	Procedure: updateStudentperExam (Update Student per Exam)	50
2.81.	Procedure: updateTopic (Update Topic)	. 50



Legend

- **?** Primary key
- Primary key disabled
- 1 User-defined primary key
- **?** Unique key
- ¶ Unique key disabled
- 1 User-defined unique key
- Active trigger
- Disabled trigger
- → Many to one relation
- $\succ_{\mathbf{i}}$ User-defined many to one relation
- ✓ User-defined one to many relation
- \succ User-defined many to many relation
- One to one relation
- user-defined one to one relation
- •**@** Input
- @ Output
- Input/Output
- Uses dependency
- User-defined uses dependency
- Used by dependency
- Tuser-defined used by dependency

Examination System

Examination-System database is designed to ensure the secure flow of the examinations' data which include all required information about the following entities: *Departments, Students, Courses, Instructors, Questions, Answers,* and the *user registration data*. It also hold tables of the relationships between the previous entities.



1. Tables

1.1. Table: Answer (Table of Answers)

Status: Active

This table will have all the details of the **Answer** entity.

Columns

		Name	Data type	Description / Attributes
■	1	Ansld	int	Answer ID
■		AnsText	varchar(30)	Answer's text
■		QID	int	ID of the answer's question References: Question

Links to

	Table	Join	Title / Name / Description
→	Question (Table of Questions)	AnswerQID = QuestionQID	FK_Answer_Question

Unique keys

_	-1 7 -		
			Name / Description
P	Ansld	PK_Answer	

Uses

0303		
	Name	
Answer (Table of Answers)		
→ Question (Table of Questions)		

Used By

Name
Answer (Table of Answers)
deleteAnswer (Delete an answer)
getAllAnswers (Retrieve all answers)
insertAnswer (Insert an answer)
updateAnswer (Update Answer)

1.2. Table: Course (Table of Courses)

Status: Active

This table will have all the details of the **Course** entity.

Columns

		Name	Data type	Description / Attributes
■	1	CrsID	int	Course ID
■		CrsName	varchar(50)	Course's name Nullable

	Name	Data type	Description / Attributes
■	InstID	int	ID of the Course's Instructor Nullable References: Instructor

Links to

	Table	Join	Title / Name / Description
—	Instructor (Table of Instructors)	CourseInstID = InstructorInstID	FK_Course_Instructor

Linked from

	Table	Join	Title / Name / Description
\rightarrow	Course_Topics (Course & Topic)	CourseCrsID = Course_TopicsCourseID	Course_Topics_FK2
\rightarrow	Dept_Course (Department & Course)	CourseCrsID = Dept_CourseCourseID	Dept_Course_FK1
\rightarrow	Question (Table of Questions)	CourseCrsID = QuestionCrsID	FK_Question_Course
\rightarrow	Stud_Course (Student & Course)	CourseCrsID = Stud_CourseCourseID	Stud_Course_FK2

Unique keys

			Name / Description
Ŷ	CrsID	PK_Course	

Uses

	Name	
E Course (Table of Courses)		
→ Instructor (Table of Instructors)		

Used By

Name
E Course (Table of Courses)
deleteCourse (Delete a course)
getAllCourses (Retrieve all Courses)
insertCourse (Insert a course)
NCourse_NumStud
🔯 stud_grade
updateCourse (Update Course)
→ Course_Topics (Course & Topic)
→ Dept_Course (Department & Course)
→ Question (Table of Questions)
→ Stud_Course (Student & Course)

1.3. Table: Course_Topics (Course & Topic)

Status: Active

This table will have all the details of the "Course & Topic" relationship. One (Course) to many (Topic)

Columns

		Name	Data type	Description / Attributes
■	1	TopicID	int	Course ID References: Topic
■		CourselD	int	Topic ID References: Course

Links to

	Table	Join	Title / Name / Description
-	Course (Table of Courses)	Course_TopicsCourseID = CourseCrsID	Course_Topics_FK2
>	Topic (Table of Topics)	Course_TopicsTopicID = TopicTID	Course_Topics_FK1

Unique keys

		Name / Description
Ŷ	TopicID	Course_Topics_PK

Uses

0303	
	Name
E Course_Topics (Course & Topic)	
→ Course (Table of Courses)	
→ Topic (Table of Topics)	

Used By

	Name
	Course_Topics (Course & Topic)
1	deleteCourseTopic (Delete a topic in a course)
\$	getAllCoursesTopics (Retrieve all topics of a course)
ŧ	insertCourseTopic (Insert a topic in a course)
ŧ	topics
1	updateCourseTopic (Update Course Topic)

1.4. Table: Department (Table of departments)

Status: Active

This table will have all the details of the **Department** entity.

Columns

		Name	Data type	Description / Attributes
■	1	Deptld	int	Department ID
■		DeptName	varchar(50)	Department's name Nullable

Linked from

	Table	Join	Title / Name / Description
\rightarrow	Dept_Course (Department & Course)	DepartmentDeptId = Dept_CourseDeptID	Dept_Course_FK2
\rightarrow	Dept_Stud (Departments & Students)	DepartmentDeptId = Dept_StudDeptID	Dept_Stud_FK2

Unique keys

		Name / Description
P	Deptld	PK_Department

Used By

	Name				
==	■ Department (Table of departments)				
÷,	deleteDepartment (Delete a department)				
÷,	getAllDepartments (Retrieve all departments)				
÷,	insertDepartment (Insert a department)				
4	updateDepartment (Update Department)				
-	→ Dept_Course (Department & Course)				
_	→ Dept_Stud (Departments & Students)				

1.5. Table: Dept_Course (Department & Course)

Status: Active

This table will have all the details of the "**Department** & **Course**" relationship. Many (**Department**) to many (**Course**)

Columns

		Name	Data type	Description / Attributes
■	1	CourseID	int	Course ID References: Course
	1	DeptID	int	Department ID References: Department
		Date_Of_Insertion	datetime	Define the date when the course (CourseID) joined the department (DeptID) Nullable Default: getdate()

Links to

	Table	Join	Title / Name / Description
\rightarrow	Course (Table of Courses)	Dept_Course CourseID = CourseCrsID	Dept_Course_FK1
-	Department (Table of departments)	Dept_CourseDeptID = DepartmentDeptId	Dept_Course_FK2

		Name / Description
?	CourseID, DeptID	Dept_Course_PK

	Name		
=	☐ Dept_Course (Department & Course)		
:	→ Course (Table of Courses)		
	→ Department (Table of departments)		

Used By

	Name		
	■ Dept_Course (Department & Course)		
\$	deletedepartmentCourse (Delete a course in a department)		
\$	getAllDepartmentscourse (Retrieve all courses per department)		
\$	insertDepartmentCourse (Insert a course in a department)		
1	updateDepartmentCourse (Update Department Course)		

1.6. Table: Dept_Stud (Departments & Students)

Status: Active

This table will have all the details of the "**Department** & **Student**" relationship. One (**Department**) to many (**Student**)

Columns

		Name	Data type	Description / Attributes
■	1	StudID	int	Student ID References: Student
■	1	DeptID	int	Department ID References: Department
		Date_Of_Insertion	datetime	Define the date when the student (StudID) joined the department (DeptID) Nullable

Links to

	Table	Join	Title / Name / Description
-	Department (Table of departments)	Dept_StudDeptID = DepartmentDeptId	Dept_Stud_FK2
>	Student (Table of Students)	Dept_StudStudID = StudentSID	Dept_Stud_FK1

Unique keys

		Name / Description
Ŷ	StudID, DeptID	Dept_Stud_PK

	Name
■ Dept_Stud (Departments & Students)	
→ Department (Table of departments)	
→ Student (Table of Students)	

Used By

	Name		
=	■ Dept_Stud (Departments & Students)		
1	deleteDepartmentStudent (Delete a student in a department)		
4	getAllDepartmentsStudents (Retrieve all students per department)		
4	insertDepartmentStudent (insert a student in a student)		
1	🔯 stud_info		
4	updateDepartmentStudent (Update Department Student)		

1.7. Table: Exam (Table of Exams)

Status: Active

This table will have all the details of the ${\bf Exam}$ entity.

Columns

		Name	Data type	Description / Attributes
目	1	ExamID	int	Exam ID
■		ExamTitle	varchar(20)	Exam's title Nullable
		Duration	float	Duration of the exam in minutes Nullable
		date	datetime	Student's address Default: getdate()

Linked from

	Table	Join	Title / Name / Description
\rightarrow	Exam_Ques (Exam & Question)	Exam ExamID = Exam_QuesExamID	Exam_Ques_FK2
\rightarrow	St_exam_Q_A (Student & Exam & Question)	Exam ExamID = St_exam_Q_AExamID	St_exam_Q_A_FK2
\rightarrow	Stud_Exam (Student & Exam)	Exam ExamID = Stud_ExamExamID	Stud_Exam_FK2

Unique keys

		Name / Description
?	ExamID	PK_Exam

Used By

	Name
Ⅲ E	Exam (Table of Exams)
Š	deleteExam (Delete an exam)
Ž	ExamGeneration (Exam Generation)
Ž	getAllExams (Retrieve all exams)
\$	insertExam (Insert an exam)
*	updateExam (Update Exam)
-	← Exam_Ques (Exam & Question)

Name
→ St_exam_Q_A (Student & Exam & Question)
→ Stud_Exam (Student & Exam)

1.8. Table: Exam_Ques (Exam & Question)

Status: Active

This table will have all the details of the "**Exam** & **Question**" relationship. Many (**Exam**) to many (**Question**)

Columns

		Name	Data type	Description / Attributes
■	1	QuesID	int	Question ID References: Question
B	1	ExamID	int	Exam ID References: Exam

Links to

	Table	Join		Title / Name / Description
\rightarrow	Exam (Table of Exams)	Exam_QuesExamID = ExamExamID	Exam_Ques_FK2	
—	Question (Table of Questions)	Exam_QuesQuesID = QuestionQID	Exam_Ques_FK1	

Unique keys

	1 /				
			K	Name / Description	
9	QuesID, ExamID	Exam_Ques_PK			

Uses

	Name
Exam_Ques (Exam & Question)	
→ Exam (Table of Exams)	
→ Question (Table of Questions)	

Used By

030	
	Name
	Exam_Ques (Exam & Question)
	deleteExamQuestion (Delete a question in an exam)
	ExamGeneration (Exam Generation)
	getAllExamsQuestions (Retrieve all questions per exam)
	insertExamQuestion (Insert a question in an exam)
	uestions questions
	updateExamQuestion (Update Exam Question)

1.9. Table: Instructor (Table of Instructors)

Status: Active

This table will have all the details of the **Instructor** entity.

Columns

		Name	Data type	Description / Attributes
■	1	InstID	int	Instructor ID
■		fname	varchar(20)	Instructor's first name
■		Iname	varchar(20)	Instructor's last name
■		age	int	Instructor's age Nullable
B		address	varchar(30)	Instructor's address Nullable

Linked from

	Table	Join	Title / Name / Description
\rightarrow	Course (Table of Courses)	InstructorInstID = CourseInstID	FK_Course_Instructor
\rightarrow	Regis_Inst (Registrar & Instructor)	InstructorInstID = Regis_InstInstID	Regis_Inst_FK1

Unique keys

		Name / Description
9	InstID	PK_Instructor

Used By

	Name
■	Instructor (Table of Instructors)
t	deleteInstructor (Delete an instructor)
t	getAllInstructors (Retrieve all instructors)
1	insertInstructor (Insert an instructor)
1	updateInstructor (Update Instructor)
-	← Course (Table of Courses)
	← Regis_Inst (Registrar & Instructor)

1.10. Table: Question (Table of Questions)

Status: Active

This table will have all the details of the **Question** entity.

Columns

		Name	Data type	Description / Attributes
■	1	QID	int	Question ID
■		QuesText	varchar(200)	Question's text
■		Туре	varchar(10)	It only expects values of "MCQ" or "T/F" Default: 'MCQ'
■		ModelAns	varchar(30)	Question's model answer.
		CrsID	int	ID of the question's course Nullable References: Course

	Name	Data type	Description / Attributes
■	advLevel	varchar(50)	Defines the level of difficulty of the question. Nullable

Links to

	Table	Join	Title / Name / Description
>	Course (Table of Courses)	QuestionCrsID = CourseCrsID	FK_Question_Course

Linked from

	Table	Join	Title / Name / Description
\rightarrow	Answer (Table of Answers)	QuestionQID = AnswerQID	FK_Answer_Question
\rightarrow	Exam_Ques (Exam & Question)	QuestionQID = Exam_QuesQuesID	Exam_Ques_FK1
\rightarrow	St_exam_Q_A (Student & Exam & Question)	QuestionQID = St_exam_Q_AQuesID	St_exam_Q_A_FK3

Unique keys

		Name / Description
Ŷ	QID	PK_Question

Uses

USES	
	Name
□ Question (Table of Questions)	
→ Course (Table of Courses)	

Used By

Name
Question (Table of Questions)
deleteQuestion (Delete a question)
examCorrection (Exam Correction)
getAllQuestions (Retrieve all questions)
insertQuestion (Insert a question)
questions
auestions_studAnswer
updateQuestion (Update Question)
→ Answer (Table of Answers)
→ Exam_Ques (Exam & Question)
→ St_exam_Q_A (Student & Exam & Question)

1.11. Table: Regis_Inst (Registrar & Instructor)

Status: Active

This table will have all the details of the "Registrar & Instructor" relationship. One (Registrar) to one (Instructor)

Columns

		Name	Data type	Description / Attributes
	1	InstID	int	Instructor ID References: Instructor
■	1	RegisID	int	User ID References: Registrar
B		Date_Of_Insertion	datetime	Define the date of registration Nullable

Links to

	Table	Join	Title / Name / Description
\rightarrow	Instructor (Table of Instructors)	Regis_Inst D = InstructorInst D	Regis_Inst_FK1
>	Registrar (Table of users information (Registeration))	Regis_InstRegisID = RegistrarRegID	Regis_Inst_FK2

Unique keys

	- · · · · · · · · · · · · · · · · · · ·		
		Name / Description	
Ŷ	InstID, RegisID	Regis_Inst_PK	

Uses

0363	
	Name
■ Regis_Inst (Registrar & Instructor)	
→ Instructor (Table of Instructors)	
→ Registrar (Table of users information (Registeration))	

Used By

Name
Regis_Inst (Registrar & Instructor)
deleteRegisterInstructor (delete an instructor user)
getAllRegisterInstructors (Retrieve all instructors and their the user ids)
insertRegisterInstructor (Insert an Instructor user)
updateRegisterInstructor (Update Registered Instructor)

1.12. Table: Regis_Stud (Registrar & Student)

Status: Active

This table will have all the details of the "Registrar & Student" relationship. One (Registrar) to one (Student)

Columns

		Name	Data type	Description / Attributes
■	1	StudId	int	Student ID References: Student
■	1	RegisID	int	User ID References: Registrar
B		Date_Of_Insertion	datetime	Define the date of registration Nullable

Links to

	Table	Join	Title / Name / Description
>	Registrar (Table of users information (Registeration))	Regis_StudRegisID = RegistrarRegID	Regis_Stud_FK2
—	Student (Table of Students)	Regis_StudStudId = StudentSID	Regis_Stud_FK1

Unique keys

		Name / Description
Ŷ	StudId, RegisID	Regis_Stud_PK

Uses

Name
III Regis_Stud (Registrar & Student)
→ Registrar (Table of users information (Registeration))
→ Student (Table of Students)

Used By

J	
	Name
■ Regis_Stud (Registrar & Student)	
deleteRegisterStudent (Delete a student user)	
getAllRegisterStudents (Retrieve all students and their the user ic	(s)
insertRegisterStudent (Insert a student user)	
updateRegisterStudent (Update Registered Student)	

1.13. Table: Registrar (Table of users information (Registeration))

Status: Active

This table will have all the details of the **Registrar** entity.

Columns

		Name	Data type	Description / Attributes	
I	1	♀ RegID	int	Registrar ID	
ŀ		Email	varchar(50)	User's email address	
ŀ		username	varchar(50)	Username	
-		password	nchar(50)	Password	
		usertype	varchar(20)	Type of the user, it should be "student" or "instructor" Nullable	

Linked from

	Table	Join	Title / Name / Description
\rightarrow	Regis_Inst (Registrar & Instructor)	RegistrarRegID = Regis_InstRegisID	Regis_Inst_FK2
\rightarrow	Regis_Stud (Registrar & Student)	RegistrarRegID = Regis_StudRegisID	Regis_Stud_FK2

Unique keys

		Name / Description
?	RegID	PK_Registrar

Used By

Name		
Registrar (Table of users information (Registeration))		
deleteRigstrar (delete user information)		
🗽 getAllRegistrars (Retrieve all user's info)		
insertRegistrar (Insert a user's info)		
→ Regis_Inst (Registrar & Instructor)		
→ Regis_Stud (Registrar & Student)		

1.14. Table: St_exam_Q_A (Student & Exam & Question)

Status: Active

This table will have all the details of the "**Student** & **Exam** & **Question**" relationship. Many (**Student**) to many (**Exam**) to Many (**Question**)

Columns

		Name	Data type	Description / Attributes
■	1	StudID	int	Student ID References: Student
	1	ExamID	int	Exam ID References Exam
■	1	QuesID	int	Question ID References: Question
		Grade	int	Student(StudID)'s grade in question(QuesID) in exam (ExamID) Nullable
		Answer	varchar(50)	Student(StudID)'s answer in question(QuesID) in exam (ExamID) Nullable

Links to

	Table	Join	Title / Name / Description
>	Exam (Table of Exams)	St_exam_Q_AExamID = ExamExamID	St_exam_Q_A_FK2
>	Question (Table of Questions)	St_exam_Q_AQuesID = QuestionQID	St_exam_Q_A_FK3
>	Student (Table of Students)	St_exam_Q_AStudID = StudentSID	St_exam_Q_A_FK1

		Name / Description
Ŷ	StudID, ExamID, QuesID	St_exam_Q_A_PK

	Name
→ Exam (Table of Exams)	
→ Question (Table of Questions)	
→ Student (Table of Students)	

Used By

	Name		
=			
1	deleteStudentExamQuestionGradeAnswer (Delete a row in St_exam_Q_A table)		
1	examAnswer (Student' Answers of the exam)		
1	examCorrection (Exam Correction)		
1	getAllStudentsExamsQuestionsGradesAnswers (Retrieve all exams' answers for students)		
1	insertStudentExamQuestionGradeAnswer (Insert a student's answer in an Exam's Question)		
1	questions_studAnswer		
1	updateStudentExamQuestionGradeAnswer (Update Student Exam Question Grade Answer)		

1.15. Table: Stud_Course (Student & Course)

Status: Active

This table will have all the details of the "**Student** & **Course**" relationship. Many (**Course**) to many (**Student**)

Columns

		Name	Data type	Description / Attributes
■	1	StudID	int	Student ID References: Student
■	1	CourseID	int	Course ID References: Course
■		FullGrade	int	Student(StudID)'s overall grade in course (CourseID) Nullable
B		Progress	varchar(50)	Define the status of the student(StudID) in the course(CourseID) Nullable

Links to

	Table	Join	Title / Name / Description
—	Course (Table of Courses)	Stud_CourseCourseID = CourseCrsID	Stud_Course_FK2
—	Student (Table of Students)	Stud_CourseStudID = StudentSID	Stud_Course_FK1

		Name / Description
P	StudID, CourseID	Stud_Course_PK

	Name	
= 9	Stud_Course (Student & Course)	
>	— Course (Table of Courses)	
>	— Student (Table of Students)	

Used By

	Name	
==	Stud_Course (Student & Course)	
1	deleteStudentperCourse (Delete student per course)	
4	getAllStudentsperCourses (Retrieve all students per courses)	
1	insertStudentperCourse (Insert a student per course)	
4	NCourse_NumStud	
4	stud_grade	
4	updateStudentperCourse (Update Student per Course)	

1.16. Table: Stud_Exam (Student & Exam)

Status: Active

This table will have all the details of the "**Student** & **Exam**" relationship. Many (**Student**) to many (**Exam**)

Columns

		Name	Data type	Description / Attributes
■	1	StudID	int	Student ID References: Student
■	1	ExamID	int	Exam ID References: Exam
■		Grade	int	Student(StudID)'s grade in exam (ExamID) Nullable
		Date_Of_Insertion	datetime	Define the date when the student (StudID) took/ will take the exam(ExamID) Nullable

Links to

	Table	Join	Title / Name / Description
\rightarrow	Exam (Table of Exams)	Stud_ExamExamID = ExamExamID	Stud_Exam_FK2
—	Student (Table of Students)	Stud_ExamStudID = StudentSID	Stud_Exam_FK1

		Name / Description
?	StudID, ExamID	Stud_Exam_PK

		
	Name	
	Stud_Exam (Student & Exam)	
	→ Exam (Table of Exams)	
	→ Student (Table of Students)	

Used By

	Name	
stud_Exam (Student & Exam)		
deleteStudentperExam (Delete student per exam)		
getAllStudentsperExams (Retrieve all students per exam)		
insertStudentperExam (Insert Student per Exam)	insertStudentperExam (Insert Student per Exam)	
updateStudentperExam (Update Student per Exam)		

1.17. Table: Student (Table of Students)

Status: Active

This table will have all the details of the **Student** entity.

Columns

		Name	Data type	Description / Attributes	
■	1	SID	int	Student ID	
■		fname	varchar(20)	Student's first name	
■		Iname	varchar(20)	Student's last name	
■		age	int	Student's age Nullable	
B		address	varchar(30)	Student's address Nullable	

Linked from

	Table Join		Title / Name / Description	
\rightarrow	Dept_Stud (Departments & Students)	StudentSID = Dept_StudStudID	Dept_Stud_FK1	
\rightarrow	Regis_Stud (Registrar & Student)	StudentSID = Regis_StudStudId	Regis_Stud_FK1	
\rightarrow	St_exam_Q_A (Student & Exam & Question)	Student SID = St_exam_Q_AStudID	St_exam_Q_A_FK1	
\rightarrow	Stud_Course (Student & Course)	StudentSID = Stud_CourseStudID	Stud_Course_FK1	
\rightarrow	Stud_Exam (Student & Exam)	StudentSID = Stud_ExamStudID	Stud_Exam_FK1	

		Name / Description
?	SID	PK_Student

Used By

	Name
	Student (Table of Students)
	deleteStudent (Delete a student)
	getAllStudents (Retrieve all students)
	insertStudent (Insert a student)
	stud_info
	updateStudent (Update Student)
	→ Dept_Stud (Departments & Students)
	→ Regis_Stud (Registrar & Student)
	→ St_exam_Q_A (Student & Exam & Question)
	→ Stud_Course (Student & Course)
	→ Stud_Exam (Student & Exam)

1.18. Table: Topic (Table of Topics)

Status: Active

This table will have all the details of the **Topic** entity.

Columns

		Name	Data type	Description / Attributes
■	1	TID	int	Topic ID
■		TopName	varchar(100)	Topic's name Nullable

Linked from

	Table	Join	Title / Name / Description
\rightarrow	Course_Topics (Course & Topic)	Topic TID = Course_TopicsTopicID	Course_Topics_FK1

Unique keys

		Name / Description
Ŷ	TID	PK_Topic

Used By

	Name
Ⅲ Topic (T	able of Topics)
🌣 delete	eTopic (Delete a topic)
🌣 getAl	ITopics (Retreive all Topics)
insert 🌣	Topic (Insert topic)
topics	
🌣 upda	teTopic (Update Topic)
→ Cours	se_Topics (Course & Topic)

2. Procedures

2.1. Procedure: deleteAnswer (Delete an answer)

Status: Active

Stored procedured for managing deleting any row in **Answer** table

Input/Output

	Name	Data type	Description
→ @	Ans_ld	int	Answer ID

Uses

	Name	
deleteAnswer (Delete an answer)		
Answer (Table of Answers)		

2.2. Procedure: deleteCourse (Delete a course)

Status: Active

Stored procedured for managing deleting any row in **Course** table

Input/Output

	Name	Data type	Description
→ @	ConditionValue	int	Course ID

Uses

				Name	
*	deleteCourse (Delete a course)				
	Course (Table of Courses)				

2.3. Procedure: deleteCourseTopic (Delete a topic in a course)

Status: Active

Stored procedured for managing deleting any row in **Course_Topics** table

Input/Output

	Name	Data type	Description
→ @	topic_ID	int	Topic ID

	Name		
ıÇı	deleteCourseTopic (Delete a topic in a course)		
1	■ Course_Topics (Course & Topic)		

2.4. Procedure: deleteDepartment (Delete a department)

Status: Active

Stored procedured for managing deleting any row in **Department** table

Input/Output

	Name	Data type	Description
→ @	ConditionValue	int	Department ID

Uses

	Name	
ı¢:	deleteDepartment (Delete a department)	
	Department (Table of departments)	

2.5. Procedure: deletedepartmentCourse (Delete a course in a department)

Status: Active

Stored procedured for managing deleting any row in **Dept_Course** table

Input/Output

	Name	Data type	Description
→@	COURSE_ID	int	Course ID
→ @	DEPT_ID	int	Department ID

Uses

	Name
deletedepartmentCourse (Delete a course in a department	
Dept_Course (Department & Course)	

2.6. Procedure: deleteDepartmentStudent (Delete a student in a department)

Status: Active

Stored procedured for managing deleting any row in **Dept_Stud** table

Input/Output

	Name	Data type	Description
→ @	STUDENT_ID	int	Student ID
→ @	DEPARTMENT_ID	int	Department ID

	Name	
*	deleteDepartmentStudent (Delete a student in a department)	
Ę.	Dept_Stud (Departments & Students)	

2.7. Procedure: deleteExam (Delete an exam)

Status: Active

Stored procedured for managing deleting any row in **Exam** table

Input/Output

	Name	Data type	Description
→ @	ExamID	int	Exam ID

Uses

	Name		
ıÇı	deleteExam (Delete an exam)		
1	Exam (Table of Exams)		

2.8. Procedure: deleteExamQuestion (Delete a question in an exam)

Status: Active

Stored procedured for managing deleting any row in **Exam_Ques** table

Input/Output

	Name	Data type	Description
→@	EID	int	Exam ID
→ @	QID	int	Question ID

Uses

	Name
deleteExamQuestion (Delete a question in an exam)	
Exam_Ques (Exam & Question)	

2.9. Procedure: deleteInstructor (Delete an instructor)

Status: Active

Stored procedured for managing deleting any row in Instructor table

Input/Output

	Name	Data type	Description
→ @	Inst_ID	int	Instructor ID

	Name		
:\$1	deleteInstructor (Delete an instructor)		
Ų Ē	Instructor (Table of Instructors)		

2.10. Procedure: deleteQuestion (Delete a question)

Status: Active

Stored procedured for managing deleting any row in **Question** table

Input/Output

	Name	Data type	Description
→ @	Q_ID	int	Question ID

Uses

	Name			
ıţ:	deleteQuestion (Delete a question)			
Ų Ē	Question (Table of Questions)			

2.11. Procedure: deleteRegisterInstructor (delete an instructor user)

Status: Active

Stored procedured for managing deleting any row in Regis_Inst table

Input/Output

	Name	Data type	Description
→ @	InstID	int	Instructor ID
→ @	registerID	int	User ID in Registrar Table

Uses

	Name
deleteRegisterInstructor (delete an instructor user)	
Regis_Inst (Registrar & Instructor)	

2.12. Procedure: deleteRegisterStudent (Delete a student user)

Status: Active

Stored procedured for managing deleting any row in Regis_Stud table

Input/Output

	Name	Data type	Description
→ @	StudId	int	Student ID
→@	Regld	int	User Id in registrar table

	Name			
:\$1	deleteRegisterStudent (Delete a student user)			
Ę.	Regis_Stud (Registrar & Student)			

2.13. Procedure: deleteRigstrar (delete user information)

Status: Active

Stored procedured for managing deleting any row in Registrar table

Input/Output

	Name	Data type	Description
→ @	Reg_ID	int	User ID

Uses

	Name
deleteRigstrar (delete user information)	
₩ Regist	rar (Table of users information (Registeration))

2.14. Procedure: deleteStudent (Delete a student)

Status: Active

Stored procedured for managing deleting any row in **Student** table

Input/Output

	Name	Data type	Description
→ @	ConditionValue	int	Student ID

Uses

	Name	
deleteStudent (Delete a student)		
Student (Table of Students)		

2.15. Procedure: deleteStudentExamQuestionGradeAnswer (Delete a row in St_exam_Q_A table)

Status: Active

Stored procedured for managing deleting any row in **St_exam_Q_A** table

Input/Output

	Name	Data type	Description
→ @	SID	int	Student ID
→ @	EID	int	Exam ID
→ @	QID	int	Question ID

	Name		
:\$1	deleteStudentExamQuestionGradeAnswer (Delete a row in St_exam_Q_A table)		
Ų. E	St_exam_Q_A (Student & Exam & Question)		

2.16. Procedure: deleteStudentperCourse (Delete student per course)

Status: Active

Stored procedured for managing deleting any row in **Stud_Course** table

Input/Output

	Name	Data type	Description
→ @	SID	int	Student ID
→ @	CID	int	Course Id

Uses

	Name
deleteStudentperCourse (Delete student per course)	
Stud_Course (Student & Course)	

2.17. Procedure: deleteStudentperExam (Delete student per exam)

Status: Active

Stored procedured for managing deleting any row in **Stud_Exam** table

Input/Output

	Name	Data type	Description
→@	SID	int	Student ID
→ @	EID	int	Exam ID

Uses

0363	
	Name
deleteStudentperExam (Delete student per exam)	
Stud_Exam (Student & Exam)	

2.18. Procedure: deleteTopic (Delete a topic)

Status: Active

Stored procedured for managing deleting any row in **Topic** table

Input/Output

	Name	Data type	Description
→ @	ConditionValue	int	Topic ID

	Name
*	deleteTopic (Delete a topic)
Ĺ	Topic (Table of Topics)

2.19. Procedure: examAnswer (Student' Answers of the exam)

Status: Active

Stored procedured for managing storing student answer in the exam (inserting one answer per running)

Input/Output

	Name	Data type	Description
→ @	student_ld	int	Student ID
→ @	exam_ld	int	Exam ID
→ @	question_ID	int	Question ID
→ @	Student_Answer	varchar(50)	Student 's answer

Uses

	Name	
examAnswer (Student' Answers of the exam)		
St_exam_Q_A (Student & Exam & Question)		

2.20. Procedure: examCorrection (Exam Correction)

Status: Active

Stored procedure that is responsible for correcting student's answers that has been collected and stored by the stored procedure **examAnswer**

Input/Output

	Name	Data type	Description
→ @	examID	int	Exam ID
→ @	studentID	int	Student ID

Uses

Name
examCorrection (Exam Correction)
Question (Table of Questions)
St_exam_Q_A (Student & Exam & Question)

2.21. Procedure: examGeneration (Exam Generation)

Status: Active

Stored procedure that is responsible for generating random question given the number of the question of each type (MCQ or True/False)

Input/Output

	Name	Data type	Description
→ @	courseID	int	Course ID
→ @	tfNum	int	Number of True/ False Questions
→ @	mcqNum	int	Number of MCQ Questions

	Name
\$	examGeneration (Exam Generation)
Ų E	Exam (Table of Exams)
Ë	Exam_Ques (Exam & Question)

2.22. Procedure: getAllAnswers (Retrieve all answers)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Answers** table

Uses

	Name
*	getAllAnswers (Retrieve all answers)
	Answer (Table of Answers)

2.23. Procedure: getAllCourses (Retrieve all Courses)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Course** table

Uses

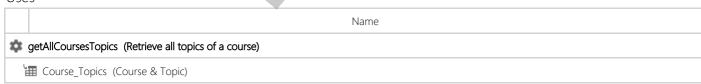


2.24. Procedure: getAllCoursesTopics (Retrieve all topics of a course)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Course_topics** table

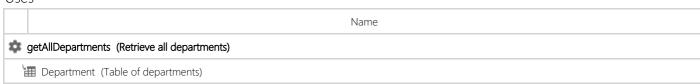
Uses



2.25. Procedure: getAllDepartments (Retrieve all departments)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Department** table

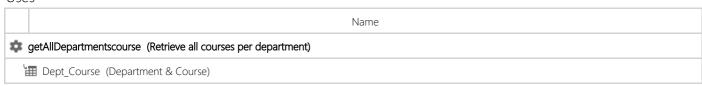


2.26. Procedure: getAllDepartmentscourse (Retrieve all courses per department)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Dept_Course** table

Uses



2.27. Procedure: getAllDepartmentsStudents (Retrieve all students per department)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Dept_Stud** table

Uses



2.28. Procedure: getAllExams (Retrieve all exams)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Exam** table

Uses

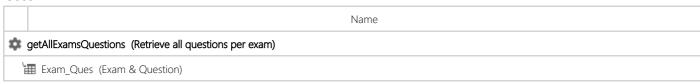


2.29. Procedure: getAllExamsQuestions (Retrieve all guestions per exam)

Status: Active

Stored procedure that is responsible for retrieving all the records in the Exam_Ques table

Uses

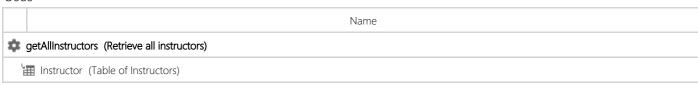


2.30. Procedure: getAllInstructors (Retrieve all instructors)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Instructor** table

н	- 1	~	-
	- 1	\sim	(



2.31. Procedure: getAllQuestions (Retrieve all questions)

Status: Active

Stored procedure that is responsible for retrieving all the records in the Question table





2.32. Procedure: getAllRegisterInstructors (Retrieve all instructors and their the user ids)

Status: Active

Stored procedure that is responsible for retrieving all the records in the Regis_Inst table





2.33. Procedure: getAllRegisterStudents (Retrieve all students and their the user ids)

Status: Active

Stored procedure that is responsible for retrieving all the records in the Regis_Studtable

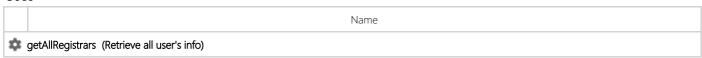
Uses



2.34. Procedure: getAllRegistrars (Retrieve all user's info)

Status: Active

Stored procedure that is responsible for retrieving all the records in the Registrar table



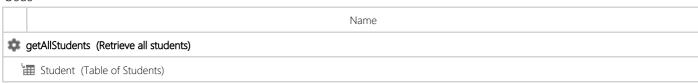
Name
 Registrar (Table of users information (Registeration))

2.35. Procedure: getAllStudents (Retrieve all students)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Student** table

Uses

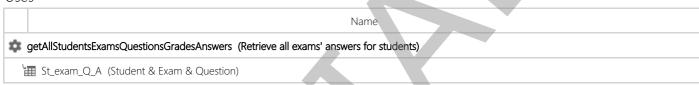


2.36. Procedure: getAllStudentsExamsQuestionsGradesAnswers (Retrieve all exams' answers for students)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **St_exam_Q_A** table

Uses

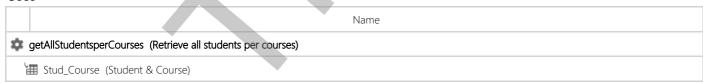


2.37. Procedure: getAllStudentsperCourses (Retrieve all students per courses)

Status: Active

Stored procedure that is responsible for retrieving all the records in the Stud_Course table

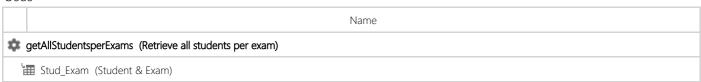
Uses



2.38. Procedure: getAllStudentsperExams (Retrieve all students per exam)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Stud_Exam** table



2.39. Procedure: getAllTopics (Retreive all Topics)

Status: Active

Stored procedure that is responsible for retrieving all the records in the **Topic** table

Uses

0000			
	Name		
s getAll	Topics (Retreive all Topics)		
⊞ То	pic (Table of Topics)		

2.40. Procedure: insertAnswer (Insert an answer)

Status: Active

Stored procedure that is responsible for storing a record in the answer table

Input/Output

	Name	Data type	Description
→ @	Ans_ld	int	Answer ID
→ @	Ans_Text	varchar(30)	Answet text
→ @	Q_ID	int	Question ID

Uses

	Name
insertAnswer (Insert an answer)	
answer (Table of Answers)	

2.41. Procedure: insertCourse (Insert a course)

Status: Active

Stored procedure that is responsible for storing a record in the **Course** table

Input/Output

	Name	Data type	Description
→ @	Course_Id	int	Course's ID
→ @	Course_Name	varchar(50)	Course's name
→ @	Instructor_ID	int	Course's Instructor

Uses

	Name
:\$1	insertCourse (Insert a course)
Ĺ	Ⅲ Course (Table of Courses)

2.42. Procedure: insertCourseTopic (Insert a topic in a course)

Status: Active

	Name	Data type	Description
→ @	TOPIC_ID	int	Topic ID
→ @	COURSE_ID	int	Course ID

Uses

Name	
insertCourseTopic (Insert a topic in a course)	
Course_Topics (Course & Topic)	

2.43. Procedure: insertDepartment (Insert a department)

Status: Active

Stored procedure that is responsible for storing a record in the **Department** table

Input/Output

	Name	Data type	Description
→ @	Department_Id	int	Department ID
→ @	Department_Name	varchar(50)	Department name

Uses

	Name	
insertDepartment (Insert a department)		
Table of departments)		

2.44. Procedure: insertDepartmentCourse (Insert a course in a department)

Status: Active

Input/Output

	Name	Data type	Description
→ @	COURSE_ID	int	Course ID
→ @	DEPT_ID	int	Department ID
→ @	INSERTDATE	datetime	Date of data insertion

Uses

	Name
*	insertDepartmentCourse (Insert a course in a department)
Ų E	■ Dept_Course (Department & Course)

2.45. Procedure: insertDepartmentStudent (insert a student in a student)

Status: Active

	Name	Data type	Description
→ @	STUDENT_ID	int	Student ID
→ @	DEPT_ID	int	Department ID
→ @	InsertionDate	datetime	Date of data insertion

Uses

	Name
:¢:	insertDepartmentStudent (insert a student in a student)
Ì	Dept_Stud (Departments & Students)

2.46. Procedure: insertExam (Insert an exam)

Status: Active

Stored procedure that is responsible for storing a record in the **Exam** table

Input/Output

	Name	Data type	Description
→ @	ExamID	int	Exam ID
→ @	ExamTitle	varchar(20)	Exam title
→ @	Duration	float	Exam duration
→ @	date	datetime	Date of data insertion

Uses

	Name	
insertExam (Insert an exam)		
Exam (Table of Exams)		

2.47. Procedure: insertExamQuestion (Insert a question in an exam)

Status: Active

Stored procedure that is responsible for storing a record in the **Exam_Ques** table

Input/Output

	Name	Data type	Description
→ @	EID	int	Exam ID
→ @	QID	int	Question ID

	Name	
*	insertExamQuestion (Insert a question in an exam)	
Ų	Exam_Ques (Exam & Question)	

2.48. Procedure: insertInstructor (Insert an instructor)

Status: Active

Stored procedure that is responsible for storing a record in the **Instructor** table

Input/Output

	Name	Data type	Description
→ @	Inst_ID	int	Instructor ID
→ @	F_name	varchar(20)	Instructor first name
→ @	L_name	varchar(20)	Instructor last name
→ @	Age	int	Instructor age
→ @	Address	varchar(30)	Instructor address

Uses

	Name	
insertInstructor (Insert an instructor)		
Instructor (Table of Instructors)		

2.49. Procedure: insertQuestion (Insert a question)

Status: Active

Stored procedure that is responsible for storing a record in the **Question** table

Input/Output

	Name	Data type	Description
→ @	Q_ID	int	Question Id
→ @	Ques_Text	varchar(200)	Question Text
→ @	type	varchar(10)	Question Type (MCQ or True/false)
→ @	Model_Ans	varchar(30)	Model answer
→@	CrsID	int	Course ID
→ @	adv_Level	varchar(50)	Question advancement level

Uses

	Name
insertQuestion (Insert a question)	
Question (Table of Questions)	

2.50. Procedure: insertRegisterInstructor (Insert an Instructor user)

Status: Active

Stored procedure that is responsible for storing a record in the **Regis_Inst** table

	Name	Data type	Description
→ @	InstID	int	Instructor ID
→ @	RegisID	int	Registration ID / User ID
→ @	Date_of_Insertion	datetime	Date of data insertion

Uses

	Name		
ф	insertRegisterInstructor (Insert an Instructor user)		
	Regis_Inst (Registrar & Instructor)		

2.51. Procedure: insertRegisterStudent (Insert a student user)

Status: Active

Stored procedure that is responsible for storing a record in the **Regis_stud** table

Input/Output

	Name	Data type	Description
→@	StudId	int	Student ID
→@	RegisID	int	Registrar ID
→@	Date_Of_Insertion	datetime	Date of data insertion

Uses

	Name	
insertRegisterStudent (Insert a student user)		
Regis_Stud (Registrar & Student)		

2.52. Procedure: insertRegistrar (Insert a user's info)

Status: Active

Stored procedure that is responsible for storing a record in the **Registrar** table

Input/Output

	Name	Data type	Description
→ @	Reg_ID	int	User ID
→ @	email	varchar(50)	User Email
→ @	uname	varchar(50)	Username
→ @	pass	nchar(50)	user's password
→ @	utype	varchar(20)	user type (student / instructor)

	Name
insertRegistrar (Insert a user's info)	

	Name
Registrar (Table of users information (Registeration))	

2.53. Procedure: insertStudent (Insert a student)

Status: Active

Stored procedure that is responsible for storing a record in the **Student** table

Input/Output

	Name	Data type	Description
→ @	Student_Id	int	Student ID
→ @	First_Name	varchar(20)	Student first name
→ @	Last_Name	varchar(20)	Student last name
→ @	Student_Age	int	Student age
→ @	Student_Address	varchar(30)	Student address

Uses

	Name
insertStudent (Insert a student)	
Student (Table of Students)	

2.54. Procedure: insertStudentExamQuestionGradeAnswer (Insert a student's answer in an Exam's Question)

Status: Active

Stored procedure that is responsible for storing a record in the **St_exam_Q_A** table

Input/Output

	Name	Data type	Description
→ @	SID	int	Student ID
→ @	EID	int	Exam Id
→@	QID	int	Question ID
→@	Qgrade	int	Grade of the question
→@	answer	varchar(50)	Student answer

	Name		
*	insertStudentExamQuestionGradeAnswer (Insert a student's answer in an Exam's Question)		
Ų E	St_exam_Q_A (Student & Exam & Question)		

2.55. Procedure: insertStudentperCourse (Insert a student per course)

Status: Active

Stored procedure that is responsible for storing a record in the **Stud_Course** table

Input/Output

	Name	Data type	Description
→ @	SID	int	Student ID
→ @	CID	int	Course ID
→ @	fgrade	int	Student Full grade in the course
→ @	progress	varchar(50)	Student Status in the course

Uses

	Name
insertStudentperCourse (Insert a student per course)	
Stud_Course (Student & Course)	

2.56. Procedure: insertStudentperExam (Insert Student per Exam)

Status: Active

Stored procedure that is responsible for storing a record in the **Stud_Exam** table

Input/Output

	Name	Data type	Description
→ @	SID	int	Student ID
→ @	EID	int	Exam ID
→ @	grade	int	Student Full Grade
→ @	date	datetime	Date of data insertion

Uses

		Name	
insertStudentperExam (Insert Student per Exam)			
⊞ Stud_Exam (Student & Exam)			

2.57. Procedure: insertTopic (Insert topic)

Status: Active

Stored procedure that is responsible for storing a record in the **Topic** table

	Name	Data type	Description
→ @	Topic_ld	int	Topic ID
→ @	Topic_Name	varchar(100)	Topic Name

	Name		
\$	insertTopic (Insert topic)		
Ĺ	Topic (Table of Topics)		

2.58. Procedure: NCourse_NumStud

Input/Output

	Name	Data type	Description
→@	InstID	int	Instructor ID (Used for Reports)

Uses

	Name
* NCourse_NumStud	
Course (Table of Courses)	_
Stud_Course (Student & Course)	

2.59. Procedure: questions

Input/Output

	Name	Data type	Description
→ @	ExamID	int	Exam ID (Used for Reports)

Uses

	Name
questions	
Exam_Ques (Exam & Question)	
☐ Question (Table of Questions)	

2.60. Procedure: questions_studAnswer

Input/Output

	Name	Data type	Description
→@	ExamID	int	Exam ID (Used for Reports)
→@	StudID	int	Student ID (Used for Reports)

Uses

	Name
:0:	questions_studAnswer
,	Question (Table of Questions)
,	St_exam_Q_A (Student & Exam & Question)

2.61. Procedure: stud_grade

	Name	Data type	Description
→ @	StudID	int	Student ID (Used for Reports)

Uses

0363	
	Name
stud_grade	
Course (Table of Courses)	
₩ Stud_Course (Student & Course)	

2.62. Procedure: stud_info

Input/Output

	Name	Data type	Description
→ @	DeptID	int	Department ID (Used for Reports)

Uses

2.63. Procedure: topics

Input/Output

	Name	Data type	Description
→ @	CoureID	int	Course ID (Used for Reports)

Uses

	Name
topics	
Example 2 Topics (Course & Topic)	
Topic (Table of Topics)	

2.64. Procedure: updateAnswer (Update Answer)

Status: Active

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(30)	The new value assigned to the edited column
→ @	ConditionValue	varchar(20)	The simple Primary key related to the desired edited row (Answer ID)

	Name
:0:	updateAnswer (Update Answer)
	answer (Table of Answers)

2.65. Procedure: updateCourse (Update Course)

Status: Active

Input/Output

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific rowDesired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(50)	The new value assigned to the edited column
→ @	ConditionValue	varchar(30)	The simple Primary key related to the desired edited row (Course ID)

Uses

	Name
updateCourse (Update Course)	
Course (Table of Courses)	

2.66. Procedure: updateCourseTopic (Update Course Topic)

Status: Active

Input/Output

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(20)	The new value assigned to the edited column
→@	ConditionValue	varchar(20)	The simple Primary key related to the desired edited row (Topic ID)

Uses

		Name
*	updateCourseTopic (Update Course Topic)	
Ë	Course_Topics (Course & Topic)	

2.67. Procedure: updateDepartment (Update Department)

Status: Active

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(50)	The new value assigned to the edited column
→ @	ConditionValue	varchar(30)	The simple Primary key related to the desired edited row (Department ID)

	Name
updateDepartment (Update Department)	
₩ De	epartment (Table of departments)

2.68. Procedure: updateDepartmentCourse (Update Department Course)

Status: Active

Input/Output

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(30)	The new value assigned to the edited column
→ @	ConditionValue1	varchar(20)	The first part of the composite primary key related to the desired edited row (Course ID)
→ @	ConditionValue2	varchar(20)	The second part of the composite primary key related to the desired edited row (Department ID)

Uses

	Name
updateDepartmentCourse (Update Department Course)	
Dept_Course (Department & Course)	

2.69. Procedure: updateDepartmentStudent (Update Department Student)

Status: Active

Input/Output

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(30)	The new value assigned to the edited column
→ @	ConditionValue1	varchar(20)	The first part of the composite primary key related to the desired edited row (Student ID)
→ @	ConditionValue2	varchar(20)	The second part of the composite primary key related to the desired edited row (Department ID)

Uses

	Name	
101	updateDepartmentStudent (Update Department Student)	
1	The Dept_Stud (Departments & Students)	

2.70. Procedure: updateExam (Update Exam)

Status: Active

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(30)	The new value assigned to the edited column
→ @	ConditionValue	varchar(20)	The simple Primary key related to the desired edited row (Exam ID)

Uses

	Name
	Name
updateExam (Update Exam)	
Exam (Table of Exams)	

2.71. Procedure: updateExamQuestion (Update Exam Question)

Status: Active

Input/Output

	Name	Data type	Description
→ @	columnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	newVal	varchar(30)	The new value assigned to the edited column
→ @	EID	int	The first part of the composite primary key related to the desired edited row (Exam ID)
→ @	QID	int	The second part of the composite primary key related to the desired edited row (Question ID)

Uses

0303			
		Name	
updateExamQuestion (Update Exam Question)			
Exam_Ques (Exam & Question)	V		

2.72. Procedure: updateInstructor (Update Instructor)

Status: Active

Input/Output

	· · · · · · · · · · · · · · · · · · ·		
	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(30)	The new value assigned to the edited column
→ @	ConditionValue	varchar(20)	The simple Primary key related to the desired edited row (Instructor ID)

	Name
ı¢ı	updateInstructor (Update Instructor)
Y	Instructor (Table of Instructors)

2.73. Procedure: updateQuestion (Update Question)

Status: Active

Input/Output

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(200)	The new value assigned to the edited column
→ @	ConditionValue	varchar(30)	The simple Primary key related to the desired edited row (Question ID)

Uses

	Name
:0:	updateQuestion (Update Question)
	Question (Table of Questions)

2.74. Procedure: updateRegisterInstructor (Update Registered Instructor)

Status: Active

Input/Output

	Name	Data type	Description
→@	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(50)	The new value assigned to the edited column
→ @	ConditionValue1	varchar(20)	The first part of the composite primary key related to the desired edited row (Instructor ID)
→ @	ConditionValue2	varchar(20)	The second part of the composite primary key related to the desired edited row (Register ID)

Uses

	Name
updateRegisterInstructor (Update Registered Instructor)	
Regis_Inst (Registrar & Instructor)	

2.75. Procedure: updateRegisterStudent (Update Registered Student)

Status: Active

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(50)	The new value assigned to the edited column
→ @	ConditionValue1	varchar(20)	The first part of the composite primary key related to the desired edited row (Student ID)
→ @	ConditionValue2	varchar(20)	The second part of the composite primary key related to the desired edited row (Register ID)

	Name	
*	updateRegisterStudent (Update Registered Student)	
Ě	Regis_Stud (Registrar & Student)	

2.76. Procedure: updateRegistrar (Update Registration)

Status: Active

Input/Output

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(50)	The new value assigned to the edited column
→ @	ConditionValue	varchar(20)	The simple Primary key related to the desired edited row (Register ID)

2.77. Procedure: updateStudent (Update Student)

Status: Active

Input/Output

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(30)	The new value assigned to the edited column
→ @	ConditionValue	varchar(30)	The simple Primary key related to the desired edited row (Student ID)

Uses

	Name
updateStudent (Update Student)	
₩ Student (Table of Students)	

2.78. Procedure: updateStudentExamQuestionGradeAnswer (Update Student Exam Question Grade Answer)

Status: Active

	Name	Data type	Description
→ @	columnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	newVal	varchar(50)	The new value assigned to the edited column
→ @	SID	int	The first part of the composite primary key related to the desired edited row (Student ID)
→ @	EID	int	The second part of the composite primary key related to the desired edited row (Exam ID)
→ @	QID	int	The third part of the composite primary key related to the desired edited row (Question ID)

	Name	
updateStudentExamQuestionGradeAnswer (Update Student Exam Question Grade Answer)		
St_exam_Q_A (Student & Exam & Question)		

2.79. Procedure: updateStudentperCourse (Update Student per Course)

Status: Active

Input/Output

	Name	Data type	Description
→ @	columnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	newVal	varchar(50)	The new value assigned to the edited column
→ @	SID	int	The first part of the composite primary key related to the desired edited row (Student ID)
→ @	CID	int	The second part of the composite primary key related to the desired edited row (Course ID)

Uses

	Name
updateStudentperCourse (Update Student per Course)	
stud_Course (Student & Course)	

2.80. Procedure: updateStudentperExam (Update Student per Exam)

Status: Active

Input/Output

	Name	Data type	Description
→ @	columnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	val	varchar(30)	The new value assigned to the edited column
→ @	SID	int	The first part of the composite primary key related to the desired edited row (Student ID)
→ @	EID	int	The second part of the composite primary key related to the desired edited row (Exam ID)

Uses

	Name
updateStudentperExam (Update Student per Exam)	
₩ Stud_Exam (Student & Exam)	

2.81. Procedure: updateTopic (Update Topic)

Status: Active

	Name	Data type	Description
→ @	ALiasColumnName	varchar(30)	Desired Column Name to Update its value in a specific row
→ @	ColumnNewValue	varchar(100)	The new value assigned to the edited column
→@	ConditionValue	varchar(30)	The simple Primary key related to the desired edited row (Topic ID)

0303	
	Name
updateTopic (Update Topic)	
Topic (Table of Topics)	

