none

Examination System

Data Dictionary

1/20/2021

Table of contents

Legend

- Primary key
- Primary key disabled
- User-defined primary key
- Unique key
- Unique key disabled
- User-defined unique key
- Active trigger
- Disabled trigger
- Many to one relation
- User-defined many to one relation
- One to many relation
- User-defined one to many relation
- Many to many relation
- 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
- User-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
		AnsId	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)	**Answer**QID = QuestionQID	FK_Answer_Question

Unique keys

		Name / Description
	AnsId	PK_Answer

Uses

	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
		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
	Course_Topics (Course & Topic)	CourseCrsID = Course_TopicsCourseID	Course_Topics_FK2
	Dept_Course (Department & Course)	CourseCrsID = Dept_CourseCourseID	Dept_Course_FK1
	Question (Table of Questions)	CourseCrsID = QuestionCrsID	FK_Question_Course
	Stud_Course (Student & Course)	CourseCrsID = Stud_CourseCourseID	Stud_Course_FK2

Unique keys

		Name / Description
	CrsID	PK_Course

Uses

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

Used By

	Name
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
▤	🔑	TopicID	int	Course ID **References**: Topic
▤		CourseID	int	Topic ID **References**: Course

Links to

	Table	Join	Title / Name / Description
⤚	Course (Table of Courses)	**Course_Topics**CourseID = CourseCrsID	Course_Topics_FK2
⤚	Topic (Table of Topics)	**Course_Topics**TopicID = TopicTID	Course_Topics_FK1

Unique keys

		Name / Description
🔑	TopicID	Course_Topics_PK

Uses

	Name
▦	Course_Topics (Course & Topic)
	⤚ Course (Table of Courses)
	⤚ Topic (Table of Topics)

Used By

	Name
▦	Course_Topics (Course & Topic)
	⚙ deleteCourseTopic (Delete a topic in a course)
	⚙ getAllCoursesTopics (Retrieve all topics of a course)
	⚙ insertCourseTopic (Insert a topic in a course)
	⚙ topics
	⚙ 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
▤	🔑	DeptId	int	Department ID
▤		DeptName	varchar(50)	Department's name Nullable

Linked from

	Table	Join	Title / Name / Description
—<	Dept_Course (Department & Course)	**Department**DeptId = Dept_CourseDeptID	Dept_Course_FK2
—<	Dept_Stud (Departments & Students)	**Department**DeptId = Dept_StudDeptID	Dept_Stud_FK2

Unique keys

		Name / Description
🔑	DeptId	PK_Department

Used By

	Name
▦	**Department (Table of departments)**
⚙	deleteDepartment (Delete a department)
⚙	getAllDepartments (Retrieve all departments)
⚙	insertDepartment (Insert a department)
⚙	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
▤	🔑	CourseID	int	Course ID **References**: Course
▤	🔑	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
>—	Course (Table of Courses)	**Dept_Course**CourseID = CourseCrsID	Dept_Course_FK1
>—	Department (Table of departments)	**Dept_Course**DeptID = DepartmentDeptId	Dept_Course_FK2

Unique keys

		Name / Description
🔑	CourseID, DeptID	Dept_Course_PK

Uses

	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)
	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
	🔑	StudID	int	Student ID **References**: Student
	🔑	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_Stud**DeptID = DepartmentDeptId	Dept_Stud_FK2
	Student (Table of Students)	**Dept_Stud**StudID = StudentSID	Dept_Stud_FK1

Unique keys

		Name / Description
🔑	StudID, DeptID	Dept_Stud_PK

Uses

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

Used By

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

1.7. Table: Exam (Table of Exams)

Status: Active

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

Columns

		Name	Data type	Description / Attributes
		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
	Exam_Ques (Exam & Question)	**Exam**ExamID = Exam_QuesExamID	Exam_Ques_FK2
	St_exam_Q_A (Student & Exam & Question)	**Exam**ExamID = St_exam_Q_AExamID	St_exam_Q_A_FK2
	Stud_Exam (Student & Exam)	**Exam**ExamID = Stud_ExamExamID	Stud_Exam_FK2

Unique keys

		Name / Description
	ExamID	PK_Exam

Used By

	Name
	Exam (Table of Exams)
	deleteExam (Delete an exam)
	ExamGeneration (Exam Generation)
	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
▤	🔑	QuesID	int	Question ID **References**: Question
▤	🔑	ExamID	int	Exam ID **References**: Exam

Links to

	Table	Join	Title / Name / Description
>—	Exam (Table of Exams)	**Exam_Ques**ExamID = ExamExamID	Exam_Ques_FK2
>—	Question (Table of Questions)	**Exam_Ques**QuesID = QuestionQID	Exam_Ques_FK1

Unique keys

		Name / Description
🔑	QuesID, ExamID	Exam_Ques_PK

Uses

	Name
▦	Exam_Ques (Exam & Question)
	>— Exam (Table of Exams)
	>— Question (Table of Questions)

Used By

	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)
	⚙ 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
▤	🔑	InstID	int	Instructor ID
▤		fname	varchar(20)	Instructor's first name
▤		lname	varchar(20)	Instructor's last name
▤		age	int	Instructor's age **Nullable**
▤		address	varchar(30)	Instructor's address **Nullable**

Linked from

	Table	Join	Title / Name / Description
⤙	Course (Table of Courses)	**Instructor**InstID = CourseInstID	FK_Course_Instructor
⤙	Regis_Inst (Registrar & Instructor)	**Instructor**InstID = Regis_InstInstID	Regis_Inst_FK1

Unique keys

		Name / Description
🔑	InstID	PK_Instructor

Used By

	Name
▦	**Instructor (Table of Instructors)**
	⚙ deleteInstructor (Delete an instructor)
	⚙ getAllInstructors (Retrieve all instructors)
	⚙ insertInstructor (Insert an instructor)
	⚙ 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
▤	🔑	QID	int	Question ID
▤		QuesText	varchar(200)	Question's text
▤		Type	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
—≺	Answer (Table of Answers)	QuestionQID = AnswerQID	FK_Answer_Question
—≺	Exam_Ques (Exam & Question)	QuestionQID = Exam_QuesQuesID	Exam_Ques_FK1
—≺	St_exam_Q_A (Student & Exam & Question)	QuestionQID = St_exam_Q_AQuesID	St_exam_Q_A_FK3

Unique keys

		Name / Description
🔑	QID	PK_Question

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
	⚙ questions_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
▤	🔑	InstID	int	Instructor ID **References**: Instructor
▤	🔑	RegisID	int	User ID **References**: Registrar
▤		Date_Of_Insertion	datetime	Define the date of registration **Nullable**

Links to

	Table	Join	Title / Name / Description
≻	Instructor (Table of Instructors)	**Regis_Inst**InstID = InstructorInstID	Regis_Inst_FK1
≻	Registrar (Table of users information (Registeration))	**Regis_Inst**RegisID = RegistrarRegID	Regis_Inst_FK2

Unique keys

		Name / Description
🔑	InstID, RegisID	Regis_Inst_PK

Uses

	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
▤	🔑	StudId	int	Student ID **References**: Student
▤	🔑	RegisID	int	User ID **References**: Registrar
▤		Date_Of_Insertion	datetime	Define the date of registration **Nullable**

Links to

	Table	Join	Title / Name / Description
≻—	Registrar (Table of users information (Registeration))	**Regis_Stud**RegisID = RegistrarRegID	Regis_Stud_FK2
≻—	Student (Table of Students)	**Regis_Stud**StudId = StudentSID	Regis_Stud_FK1

Unique keys

		Name / Description
🔑	StudId, RegisID	Regis_Stud_PK

Uses

	Name
▦	**Regis_Stud (Registrar & Student)**
	≻— Registrar (Table of users information (Registeration))
	≻— Student (Table of Students)

Used By

	Name
▦	**Regis_Stud (Registrar & Student)**
	⚙ deleteRegisterStudent (Delete a student user)
	⚙ getAllRegisterStudents (Retrieve all students and their the user ids)
	⚙ 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
▤	🔑	RegID	int	Registrar ID
▤		Email	varchar(50)	User's email address
▤		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
—≺	Regis_Inst (Registrar & Instructor)	**Registrar**RegID = Regis_InstRegisID	Regis_Inst_FK2
—≺	Regis_Stud (Registrar & Student)	**Registrar**RegID = 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
▤	🔑	StudID	int	Student ID **References**: Student
▤	🔑	ExamID	int	Exam ID **References**: Exam
▤	🔑	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

Unique keys

		Name / Description
🔑	StudID, ExamID, QuesID	St_exam_Q_A_PK

Uses

	Name
St_exam_Q_A (Student & Exam & Question)	
	Exam (Table of Exams)
	Question (Table of Questions)
	Student (Table of Students)

Used By

	Name
St_exam_Q_A (Student & Exam & Question)	
	deleteStudentExamQuestionGradeAnswer (Delete a row in St_exam_Q_A table)
	examAnswer (Student' Answers of the exam)
	examCorrection (Exam Correction)
	getAllStudentsExamsQuestionsGradesAnswers (Retrieve all exams' answers for students)
	insertStudentExamQuestionGradeAnswer (Insert a student's answer in an Exam's Question)
	questions_studAnswer
	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
	🔑	StudID	int	Student ID **References**: Student
	🔑	CourseID	int	Course ID **References**: Course
		FullGrade	int	Student(StudID)'s overall grade in course (CourseID) **Nullable**
		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_Course**CourseID = CourseCrsID	Stud_Course_FK2
	Student (Table of Students)	**Stud_Course**StudID = StudentSID	Stud_Course_FK1

Unique keys

		Name / Description
🔑	StudID, CourseID	Stud_Course_PK

Uses

	Name
Stud_Course (Student & Course)	
	Course (Table of Courses)
	Student (Table of Students)

Used By

	Name
Stud_Course (Student & Course)	
	deleteStudentperCourse (Delete student per course)
	getAllStudentsperCourses (Retrieve all students per courses)
	insertStudentperCourse (Insert a student per course)
	NCourse_NumStud
	stud_grade
	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
		StudID	int	Student ID **References**: Student
		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
	Exam (Table of Exams)	**Stud_Exam**ExamID = ExamExamID	Stud_Exam_FK2
	Student (Table of Students)	**Stud_Exam**StudID = StudentSID	Stud_Exam_FK1

Unique keys

		Name / Description
	StudID, ExamID	Stud_Exam_PK

Uses

	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)
	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
	🔑	SID	int	Student ID
		fname	varchar(20)	Student's first name
		lname	varchar(20)	Student's last name
		age	int	Student's age Nullable
		address	varchar(30)	Student's address Nullable

Linked from

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

Unique keys

		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
	🔑	TID	int	Topic ID
		TopName	varchar(100)	Topic's name Nullable

Linked from

	Table	Join	Title / Name / Description
	Course_Topics (Course & Topic)	TopicTID = Course_TopicsTopicID	Course_Topics_FK1

Unique keys

		Name / Description
🔑	TID	PK_Topic

Used By

	Name
▦	Topic (Table of Topics)
	deleteTopic (Delete a topic)
	getAllTopics (Retreive all Topics)
	insertTopic (Insert topic)
	topics
	updateTopic (Update Topic)
	Course_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_Id	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

Uses

	Name
deleteCourseTopic (Delete a topic in a course)	
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
⚙	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

Uses

	Name
⚙	deleteDepartmentStudent (Delete a student in a department)
	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
deleteExam (Delete an exam)	
	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

Uses

	Name
deleteInstructor (Delete an instructor)	
	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
⚙	deleteQuestion (Delete a question)
	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
→@	RegId	int	User Id in registrar table

Uses

	Name
⚙	deleteRegisterStudent (Delete a student user)
	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)
	▦ Registrar (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

Uses

	Name
⚙	deleteStudentExamQuestionGradeAnswer (Delete a row in St_exam_Q_A table)
	▦ 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

	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

Uses

	Name
⚙	deleteTopic (Delete a topic)
	▦ 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_Id	int	Student ID
→@	exam_Id	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

Uses

	Name
⚙	examGeneration (Exam Generation)
	Exam (Table of Exams)
	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

	Name
⚙	getAllCourses (Retrieve all Courses)
	Course (Table of Courses)

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

	Name
⚙	getAllCoursesTopics (Retrieve all topics of a course)
	Course_Topics (Course & Topic)

2.25. Procedure: getAllDepartments (Retrieve all departments)

Status: Active

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

Uses

	Name
⚙	getAllDepartments (Retrieve all departments)
	Department (Table of departments)

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

Name
getAllDepartmentscourse (Retrieve all courses per department)
Dept_Course (Department & Course)

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

Name
getAllDepartmentsStudents (Retrieve all students per department)
Dept_Stud (Departments & Students)

2.28. Procedure: getAllExams (Retrieve all exams)

Status: Active

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

Uses

Name
getAllExams (Retrieve all exams)
Exam (Table of Exams)

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

Status: Active

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

Uses

Name
getAllExamsQuestions (Retrieve all questions per exam)
Exam_Ques (Exam & Question)

2.30. Procedure: getAllInstructors (Retrieve all instructors)

Status: Active

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

Uses

	Name
⚙	getAllInstructors (Retrieve all instructors)
	Instructor (Table of Instructors)

2.31. Procedure: getAllQuestions (Retrieve all questions)

Status: Active

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

Uses

	Name
⚙	getAllQuestions (Retrieve all questions)
	Question (Table of Questions)

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

Uses

	Name
⚙	getAllRegisterInstructors (Retrieve all instructors and their the user ids)
	Regis_Inst (Registrar & Instructor)

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_Stud**table

Uses

	Name
⚙	getAllRegisterStudents (Retrieve all students and their the user ids)
	Regis_Stud (Registrar & Student)

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

Uses

	Name
⚙	getAllRegistrars (Retrieve all user's info)

	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

	Name
getAllStudents (Retrieve all students)	
	Student (Table of Students)

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

	Name
getAllStudentsExamsQuestionsGradesAnswers (Retrieve all exams' answers for students)	
	St_exam_Q_A (Student & Exam & Question)

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

	Name
getAllStudentsperCourses (Retrieve all students per courses)	
	Stud_Course (Student & Course)

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

Uses

	Name
getAllStudentsperExams (Retrieve all students per exam)	
	Stud_Exam (Student & Exam)

2.39. Procedure: getAllTopics (Retreive all Topics)

Status: Active

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

Uses

	Name
⚙	getAllTopics (Retreive all Topics)
	Topic (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_Id	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
⚙	insertCourse (Insert a course)
	Course (Table of Courses)

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

Status: Active

Input/Output

	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)
	▦ 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)
	▦ Dept_Course (Department & Course)

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

Status: Active

Input/Output

	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)	
	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

Uses

	Name
insertExamQuestion (Insert a question in an exam)	
	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

Input/Output

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

Uses

	Name
⚙	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)

Uses

	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

Uses

	Name
⚙	insertStudentExamQuestionGradeAnswer (Insert a student's answer in an Exam's Question)
	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

Input/Output

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

Uses

	Name
insertTopic (Insert topic)	
	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
questions_studAnswer	
	Question (Table of Questions)
	St_exam_Q_A (Student & Exam & Question)

2.61. Procedure: stud_grade

Input/Output

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

Uses

	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

	Name
stud_info	
	Dept_Stud (Departments & Students)
	Student (Table of Students)

2.63. Procedure: topics

Input/Output

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

Uses

	Name
topics	
	Course_Topics (Course & Topic)
	Topic (Table of Topics)

2.64. Procedure: updateAnswer (Update Answer)

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 (Answer ID)

Uses

	Name
⚙	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)
	Course_Topics (Course & Topic)

2.67. Procedure: updateDepartment (Update Department)

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(30)	The simple Primary key related to the desired edited row (Department ID)

Uses

	Name
updateDepartment (Update Department)	
	Department (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
updateDepartmentStudent (Update Department Student)	
	Dept_Stud (Departments & Students)

2.70. Procedure: updateExam (Update Exam)

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 (Exam ID)

Uses

	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

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

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)

Uses

	Name
updateInstructor (Update Instructor)	
	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
⚙	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

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 (Student ID)
→@	ConditionValue2	varchar(20)	The second part of the composite primary key related to the desired edited row (Register ID)

Uses

	Name
⚙	updateRegisterStudent (Update Registered Student)
	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

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)
→@	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)

Uses

	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

Input/Output

	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)

Uses

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