irDevelopers.com [irDevelopers.com]

Examination@.

Data Dictionary

2/18/2022

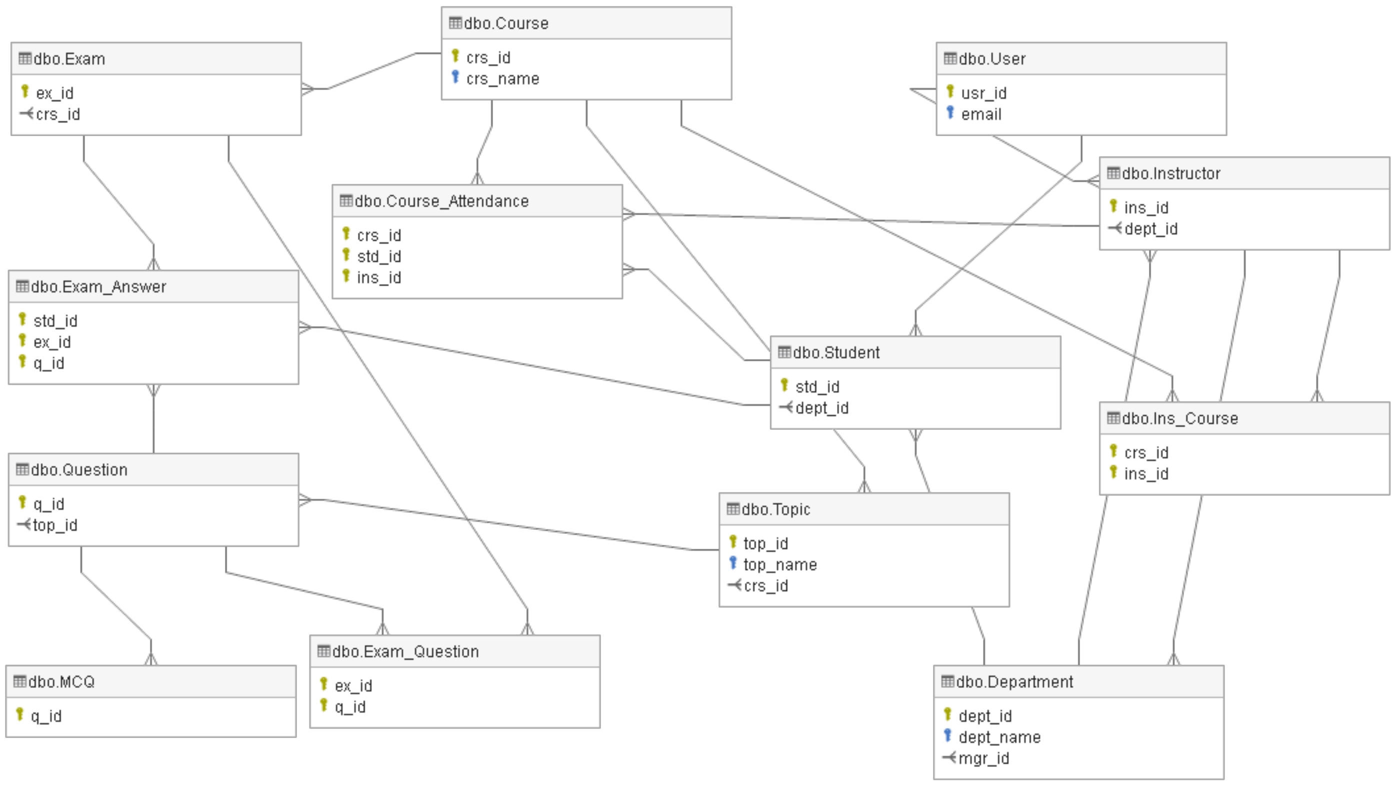
Table of contents

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
- 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@.

1. ERD



dbo.Exam
ex_id
crs_id
dbo.Course
crs_id
crs_name
dbo.User
usr_id
email
dbo.Instructor
ins_id
dept_id
dbo.Course_Attendance
crs_id
std_id
ins_id
dbo.Exam_Answer
std_id
ex_id
q_id
dbo.Student
std_id
dept_id
dbo.Ins_Course
crs_id
ins_id
dbo.Question
q_id
top_id
dbo.Topic
top_id
top_name
crs_id
dbo.MCQ
q_id
dbo.Exam_Question
ex_id
q_id
dbo.Department
dept_id
dept_name
mgr_id

2. Other

2.1. Tables

2.1.1. Table: dbo.Course

Columns

		Name	Data type	Description / Attributes
		crs_id	int	**Identity / Auto increment**
		crs_name	varchar(100)	

Linked from

	Table	Join	Title / Name / Description
	dbo.Course_Attendance	**dbo.Course**.crs_id = dbo.Course_Attendance.crs_id	FK__Course_At__crs_i__702996C1
	dbo.Exam	**dbo.Course**.crs_id = dbo.Exam.crs_id	FK__Exam__crs_id__04308F6E
	dbo.Ins_Course	**dbo.Course**.crs_id = dbo.Ins_Course.crs_id	FK__Ins_Cours__crs_i__74EE4BDE
	dbo.Topic	**dbo.Course**.crs_id = dbo.Topic.crs_id	FK__Topic__crs_id__005FFE8A

Unique keys

	Columns	Name / Description
	crs_id	PK__Course__ECAF53753732DDC6
	crs_name	UQ__Course__775BF427BE700B2C

Used By

Name
dbo.Course
dbo.Assign_Course_to_Instructor
dbo.Courses_and_Students_of_Instructor
dbo.Delete_Course
dbo.End_Course_for_Student
dbo.End_Course_with_Instructor
dbo.generateExam
dbo.Insert_Course
dbo.Insert_Topic
dbo.returnGrades
dbo.setCourseName
dbo.sp_returngrades
dbo.Student_Take_course_with_Instructor
dbo.Topics_of_Course
dbo.viewCourseMCQ
dbo.viewCourseTFQ

Name
dbo.viewMCQ
dbo.viewTFQ
dbo.viewTopicMCQ
dbo.viewTopicTFQ
dbo.Course_Attendance
dbo.Exam
dbo.Ins_Course
dbo.Topic

2.1.2. Table: dbo.Course_Attendance

Columns

		Name	Data type	Description / Attributes
		crs_id	int	**References**: dbo.Course
		std_id	int	**References**: dbo.Student
		ins_id	int	**References**: dbo.Instructor
		grade	int	**Nullable** **Computed**: ([dbo].[getStudentGrade]([crs_id],[std_id]))

Links to

	Table	Join	Title / Name / Description
	dbo.Course	**dbo.Course_Attendance**.crs_id = dbo.Course.crs_id	FK__Course_At__crs_i__702996C1
	dbo.Instructor	**dbo.Course_Attendance**.ins_id = dbo.Instructor.ins_id	FK__Course_At__ins_i__7211DF33
	dbo.Student	**dbo.Course_Attendance**.std_id = dbo.Student.std_id	FK__Course_At__std_i__711DBAFA

Unique keys

	Columns	Name / Description
	crs_id, std_id, ins_id	PK__Course_A__7D83C003E1C3556C

Uses

Name
dbo.Course_Attendance
dbo.Course_Attendance
dbo.getStudentGrade
dbo.Course
dbo.Instructor
dbo.Student

Used By

Name
dbo.Course_Attendance
dbo.Course_Attendance
dbo.Courses_and_Students_of_Instructor
dbo.End_Course_for_Student
dbo.generateExam
dbo.returnGrades
dbo.sp_returngrades
dbo.Student_Take_course_with_Instructor

2.1.3. Table: dbo.Department

Columns

		Name	Data type	Description / Attributes
		dept_id	int	**Identity / Auto increment**
		dept_name	varchar(100)	
		mgr_id	int	**References**: dbo.Instructor

Links to

	Table	Join	Title / Name / Description
	dbo.Instructor	**dbo.Department**.mgr_id = dbo.Instructor.ins_id	FK__Departmen__mgr_i__7AA72534

Linked from

	Table	Join	Title / Name / Description
	dbo.Instructor	**dbo.Department**.dept_id = dbo.Instructor.dept_id	Instructor_fk_1
	dbo.Student	**dbo.Department**.dept_id = dbo.Student.dept_id	Student_fk_1

Unique keys

	Columns	Name / Description
	dept_id	PK__Departme__DCA659742C27CF9C
	dept_name	UQ__Departme__C7D39AE1AB95A721

Uses

Name
dbo.Department
dbo.Instructor

Used By

Name
dbo.Department
dbo.v_Instructor
dbo.v_Students
dbo.Delete_Department
dbo.getDepartment
dbo.getDeptData
dbo.Insert_Department
dbo.Update_Department_Manager
dbo.Instructor
dbo.Student

2.1.4. Table: dbo.Exam

Columns

		Name	Data type	Description / Attributes
	🔑	ex_id	int	**Identity / Auto increment**
		date	date	**Default**: getdate()
		crs_id	int	**References**: dbo.Course

Links to

	Table	Join	Title / Name / Description
≻	dbo.Course	**dbo.Exam**.crs_id = dbo.Course.crs_id	FK__Exam__crs_id__04308F6E

Linked from

	Table	Join	Title / Name / Description
≺	dbo.Exam_Answer	**dbo.Exam**.ex_id = dbo.Exam_Answer.ex_id	FK__Exam_Answ__ex_id__0ADD8CFD
≺	dbo.Exam_Question	**dbo.Exam**.ex_id = dbo.Exam_Question.ex_id	FK__Exam_Ques__ex_id__070CFC19

Unique keys

	Columns	Name / Description
🔑	ex_id	PK__Exam__F6D3E48998D53083

Uses

Name
dbo.Exam
≻ dbo.Course

Used By

Name
dbo.Exam
dbo.deleteExam
dbo.generateExam
dbo.GET_QUESTIONS_for_STUDENT_EXAM
dbo.Get_Questions_in_Exam
dbo.getStudentAnswer
dbo.viewExamQuestions
fx dbo.getStudentGrade
≺ dbo.Exam_Answer
≺ dbo.Exam_Question

2.1.5. Table: dbo.Exam_Answer

Columns

		Name	Data type	Description / Attributes
		std_id	int	**References**: dbo.Student
		ex_id	int	**References**: dbo.Exam
		q_id	int	**References**: dbo.Question
		std_answer	varchar(1)	**Nullable**
		std_mark	int	**Nullable** **Computed**: ([dbo].[getQuestionMark]([q_id]))

Links to

	Table	Join	Title / Name / Description
	dbo.Exam	**dbo.Exam_Answer**.ex_id = dbo.Exam.ex_id	FK__Exam_Answ__ex_id__0ADD8CFD
	dbo.Question	**dbo.Exam_Answer**.q_id = dbo.Question.q_id	Exam_Answer_fk_1
	dbo.Student	**dbo.Exam_Answer**.std_id = dbo.Student.std_id	FK__Exam_Answ__std_i__09E968C4

Unique keys

	Columns	Name / Description
	std_id, ex_id, q_id	PK__Exam_Ans__9552224111894B0D

Uses

Name
dbo.Exam_Answer
dbo.Exam_Answer
dbo.getQuestionMark
dbo.Exam
dbo.Question
dbo.Student

Used By

Name
dbo.Exam_Answer
dbo.Exam_Answer
dbo.answerExamQuestion
dbo.deleteExam
dbo.generateExam
dbo.GET_QUESTIONS_for_STUDENT_EXAM
dbo.getStudentAnswer
dbo.getQuestionMark

Name
dbo.getStudentGrade

2.1.6. Table: dbo.Exam_Question

Columns

		Name	Data type	Description / Attributes
		ex_id	int	**References**: dbo.Exam
		q_id	int	**References**: dbo.Question

Links to

	Table	Join	Title / Name / Description
	dbo.Exam	**dbo.Exam_Question**.ex_id = dbo.Exam.ex_id	FK__Exam_Ques__ex_id__070CFC19
	dbo.Question	**dbo.Exam_Question**.q_id = dbo.Question.q_id	Exam_Question_fk_1

Unique keys

	Columns	Name / Description
	ex_id, q_id	PK__Exam_Que__E5067FB828AF0855

Uses

Name
dbo.Exam_Question
dbo.Exam
dbo.Question

Used By

Name
dbo.Exam_Question
dbo.deleteExam
dbo.generateExam
dbo.GET_QUESTIONS_for_STUDENT_EXAM
dbo.Get_Questions_in_Exam
dbo.getStudentAnswer
dbo.viewExamQuestions

2.1.7. Table: dbo.Ins_Course

Columns

		Name	Data type	Description / Attributes
		crs_id	int	**References**: dbo.Course
		ins_id	int	**References**: dbo.Instructor
		evaluation	int	**Nullable**

Links to

	Table	Join	Title / Name / Description
	dbo.Course	**dbo.Ins_Course**.crs_id = dbo.Course.crs_id	FK__Ins_Cours__crs_i__74EE4BDE
	dbo.Instructor	**dbo.Ins_Course**.ins_id = dbo.Instructor.ins_id	FK__Ins_Cours__ins_i__75E27017

Unique keys

	Columns	Name / Description
	crs_id, ins_id	c_CA_PK

Uses

Name
dbo.Ins_Course
dbo.Course
dbo.Instructor

Used By

Name
dbo.Ins_Course
dbo.Assign_Course_to_Instructor
dbo.Courses_and_Students_of_Instructor
dbo.End_Course_with_Instructor
dbo.Student_Take_course_with_Instructor

2.1.8. Table: dbo.Instructor

Columns

		Name	Data type	Description / Attributes
	🔑	ins_id	int	**References**: dbo.User
		salary	money	**Nullable**
		degree	varchar(50)	**Nullable**
		dept_id	int	**References**: dbo.Department
		hire_date	date	**Nullable** **Default**: getdate()

Links to

	Table	Join	Title / Name / Description
	dbo.Department	**dbo.Instructor**.dept_id = dbo.Department.dept_id	Instructor_fk_1
	dbo.User	**dbo.Instructor**.ins_id = dbo.User.usr_id	FK__Instructo__ins_i__6A70BD6B

Linked from

	Table	Join	Title / Name / Description
	dbo.Course_Attendance	**dbo.Instructor**.ins_id = dbo.Course_Attendance.ins_id	FK__Course_At__ins_i__7211DF33
	dbo.Department	**dbo.Instructor**.ins_id = dbo.Department.mgr_id	FK__Departmen__mgr_i__7AA72534
	dbo.Ins_Course	**dbo.Instructor**.ins_id = dbo.Ins_Course.ins_id	FK__Ins_Cours__ins_i__75E27017

Unique keys

	Columns	Name / Description
🔑	ins_id	PK__Instruct__9CB72D20AABB7C88

Uses

Name
dbo.Instructor
dbo.Department
dbo.User

Used By

Name
dbo.Instructor
dbo.v_Instructor
dbo.Assign_Course_to_Instructor
dbo.Courses_and_Students_of_Instructor
dbo.deleteInstructor
dbo.End_Course_with_Instructor

Name
dbo.Insert_Department
dbo.Insert_Department_With_Manager
dbo.Insert_Instructor
dbo.Student_Take_course_with_Instructor
dbo.updateInstructorData
dbo.Course_Attendance
dbo.Department
dbo.Ins_Course

2.1.9. Table: dbo.MCQ

Columns

		Name	Data type	Description / Attributes
		q_id	int	**References**: dbo.Question
		ch_a	varchar(300)	
		ch_b	varchar(300)	
		ch_c	varchar(300)	
		ch_d	varchar(300)	

Links to

	Table	Join	Title / Name / Description
	dbo.Question	**dbo.MCQ**.q_id = dbo.Question.q_id	FK__MCQ__q_id__155B1B70

Unique keys

	Columns	Name / Description
	q_id	PK__MCQ__3D59B3106E01F3B0

Uses

Name
dbo.MCQ
dbo.Question

Used By

Name
dbo.MCQ
dbo.deleteQuestion
dbo.Get_Questions_in_Exam
dbo.insertMCQ
dbo.updateMCQ
dbo.viewCourseMCQ
dbo.viewExamQuestions
dbo.viewMCQ
dbo.viewTopicMCQ

2.1.10. Table: dbo.Question

Columns

		Name	Data type	Description / Attributes
		q_id	int	Identity / Auto increment
		q_type	varchar(3)	
		q_text	varchar(300)	
		corr_answer	varchar(1)	
		top_id	int	**References**: dbo.Topic

Links to

	Table	Join	Title / Name / Description
	dbo.Topic	**dbo.Question**.top_id = dbo.Topic.top_id	FK__Question__top_id__0EAE1DE1

Linked from

	Table	Join	Title / Name / Description
	dbo.Exam_Answer	**dbo.Question**.q_id = dbo.Exam_Answer.q_id	Exam_Answer_fk_1
	dbo.Exam_Question	**dbo.Question**.q_id = dbo.Exam_Question.q_id	Exam_Question_fk_1
	dbo.MCQ	**dbo.Question**.q_id = dbo.MCQ.q_id	FK__MCQ__q_id__155B1B70

Unique keys

	Columns	Name / Description
	q_id	PK__Question__3D59B3106E0DF7E6

Uses

Name
dbo.Question
dbo.Topic

Used By

Name
dbo.Question
dbo.deleteQuestion
dbo.generateExam
dbo.GET_QUESTIONS_for_STUDENT_EXAM
dbo.Get_Questions_in_Exam
dbo.getStudentAnswer
dbo.updateMCQ
dbo.updateTFQ
dbo.viewCourseMCQ
dbo.viewCourseTFQ

Name
dbo.viewExamQuestions
dbo.viewMCQ
dbo.viewTFQ
dbo.viewTopicMCQ
dbo.viewTopicTFQ
PRIVATE.insertQuestion
dbo.getQuestionMark
dbo.Exam_Answer
dbo.Exam_Question
dbo.MCQ

2.1.11. Table: dbo.Student

Columns

		Name	Data type	Description / Attributes
	🔑	std_id	int	**References**: dbo.User
		dept_id	int	**References**: dbo.Department

Links to

	Table	Join	Title / Name / Description
≻	dbo.Department	**dbo.Student**.dept_id = dbo.Department.dept_id	Student_fk_1
≻	dbo.User	**dbo.Student**.std_id = dbo.User.usr_id	FK__Student__std_id__66A02C87

Linked from

	Table	Join	Title / Name / Description
≺	dbo.Course_Attendance	**dbo.Student**.std_id = dbo.Course_Attendance.std_id	FK__Course_At__std_i__711DBAFA
≺	dbo.Exam_Answer	**dbo.Student**.std_id = dbo.Exam_Answer.std_id	FK__Exam_Answ__std_i__09E968C4

Unique keys

	Columns	Name / Description
🔑	std_id	PK__Student__0B0245BA1E41A431

Uses

Name
dbo.Student
≻ dbo.Department
≻ dbo.User

Used By

Name
dbo.Student
dbo.v_Students
dbo.deleteStudent
dbo.End_Course_for_Student
dbo.generateExam
dbo.GET_QUESTIONS_for_STUDENT_EXAM
dbo.getStudentAnswer
dbo.Insert_Student
dbo.returnGrades
dbo.sp_returngrades
dbo.Student_Take_course_with_Instructor
dbo.updateStudentData

Name
dbo.Course_Attendance
dbo.Exam_Answer

2.1.12. Table: dbo.Topic

Columns

		Name	Data type	Description / Attributes
		top_id	int	**Identity / Auto increment**
		top_name	varchar(100)	
		crs_id	int	**References**: dbo.Course

Links to

	Table	Join	Title / Name / Description
	dbo.Course	**dbo.Topic**.crs_id = dbo.Course.crs_id	FK__Topic__crs_id__005FFE8A

Linked from

	Table	Join	Title / Name / Description
	dbo.Question	**dbo.Topic**.top_id = dbo.Question.top_id	FK__Question__top_id__0EAE1DE1

Unique keys

	Columns	Name / Description
	top_id	PK__Topic__B582A63DB2FB06F9
	top_name	UQ__Topic__A87EDAD67A912483

Uses

Name
dbo.Topic
dbo.Course

Used By

Name
dbo.Topic
dbo.Delete_Topic
dbo.generateExam
dbo.Insert_Topic
dbo.insertMCQ
dbo.insertTFQ
dbo.setTopicName
dbo.Topics_of_Course
dbo.updateMCQ
dbo.updateTFQ
dbo.viewCourseMCQ
dbo.viewCourseTFQ
dbo.viewMCQ
dbo.viewTFQ

Name
dbo.viewTopicMCQ
dbo.viewTopicTFQ
PRIVATE.insertQuestion
dbo.Question

2.1.13. Table: dbo.User

Columns

		Name	Data type	Description / Attributes
		usr_id	int	Identity / Auto increment
		user_type	varchar(1)	
		f_name	varchar(50)	
		l_name	varchar(50)	
		address	varchar(150)	Nullable
		email	varchar(90)	
		hashed_password	varchar(255)	

Linked from

	Table	Join	Title / Name / Description
	dbo.Instructor	dbo.User.usr_id = dbo.Instructor.ins_id	FK__Instructo__ins_i__6A70BD6B
	dbo.Student	dbo.User.usr_id = dbo.Student.std_id	FK__Student__std_id__66A02C87

Unique keys

	Columns	Name / Description
	usr_id	PK__User__60621ABCC3FFF2A9
	email	UQ__User__AB6E61646281DC68

Used By

Name
dbo.User
dbo.v_Instructor
dbo.v_Students
dbo.deleteInstructor
dbo.deleteStudent
dbo.getDepartment
dbo.getDeptData
dbo.updateUserData
PRIVATE.Insert_User
dbo.Instructor
dbo.Student

2.2. Views

2.2.1. View: dbo.v_Instructor

Columns

		Name	Data type	Description / Attributes
		usr_id	int	
		f_name	varchar(50)	
		l_name	varchar(50)	
		address	varchar(150)	Nullable
		email	varchar(90)	
		salary	money	Nullable
		degree	varchar(50)	Nullable
		dept_id	int	
		dept_name	varchar(100)	

Uses

Name
dbo.v_Instructor
dbo.Department
dbo.Instructor
dbo.User

2.2.2. View: dbo.v_Students

Columns

		Name	Data type	Description / Attributes
		usr_id	int	
		f_name	varchar(50)	
		l_name	varchar(50)	
		address	varchar(150)	Nullable
		email	varchar(90)	
		dept_id	int	
		dept_name	varchar(100)	

Uses

Name
dbo.v_Students
dbo.Department
dbo.Student
dbo.User

Used By

Name
dbo.v_Students
dbo.getAllStudents
dbo.getStudentsInDepartment

2.3. Procedures

2.3.1. Procedure: dbo.answerExamQuestion

Input/Output

	Name	Data type	Description
→@	std_id	int	
→@	ex_id	int	
→@	q_id	int	
→@	std_answer	varchar(1)	

Uses

Name
dbo.answerExamQuestion
dbo.Exam_Answer

Script

```
CREATE   PROC answerExamQuestion @std_id int, @ex_id int, @q_id int, @std_answer varchar(1)
AS
BEGIN
           Update Exam_Answer
           SET std_answer = @std_answer
           WHERE q_id = @q_id AND ex_id = @ex_id AND std_id = @std_id
END
```

2.3.2. Procedure: dbo.Assign_Course_to_Instructor

Input/Output

	Name	Data type	Description
→@	crs_name	varchar(20)	
→@	ins_id	int	

Uses

Name
dbo.Assign_Course_to_Instructor
dbo.Course
dbo.Ins_Course
dbo.Instructor

Script

```
/* ------------------------------------------------------------------------------- */
/*                  Course, Instructor CRUDs (Ins_Course table)                    */
/* ------------------------------------------------------------------------------- */

/* ------------------------------------------------------------------------------- */
/*                         Assign Instructor to Course                             */
/* ------------------------------------------------------------------------------- */

create   procedure Assign_Course_to_Instructor @crs_name varchar(20), @ins_id int
as
if exists (select crs_name from [Course] where crs_name = @crs_name)
           begin
                      if exists (select @ins_id from [Instructor] where ins_id = @ins_id)
                                 begin try
                                            insert into [Ins_Course] (crs_id, ins_id)
                                            values ((select crs_id from [Course] where crs_name = @crs_name), @ins_id)
                                 end try
                                 begin catch
                                            select 'the Instructor is already assgined to this course'
                                 end catch
                      else
                                 select 'There is no Instructor ID ' + @ins_id
           end
else
           select 'There is no Course named ' + @crs_name
```

2.3.3. Procedure: dbo.Courses_and_Students_of_Instructor

Input/Output

	Name	Data type	Description
@	ins_id	int	

Uses

Name
dbo.Courses_and_Students_of_Instructor
dbo.Course
dbo.Course_Attendance
dbo.Ins_Course
dbo.Instructor

Script

```
/* ------------------------------------------------------------------------------- */
/*    Report that takes the instructor ID and returns the name of the courses      */
/*    that he teaches and the number of student per course.                       */
/* ------------------------------------------------------------------------------- */

create   proc Courses_and_Students_of_Instructor @ins_id int
as
if exists (select ins_id from [Instructor] where ins_id = @ins_id)
           begin
                      select c.crs_name, count(ca.std_id) as 'number of students per course'
                      from Instructor i
                      inner join Ins_Course ic
                      on i.ins_id = ic.ins_id
                      inner join Course c
                      on ic.crs_id = c.crs_id
                      inner join Course_Attendance ca
                      on (c.crs_id = ca.crs_id and ca.ins_id = @ins_id)
                      where i.ins_id = @ins_id
                      group by c.crs_name
           end
else
           select CONCAT('There is no instructor with this ID ', @ins_id)
```

2.3.4. Procedure: dbo.Delete_Course

Input/Output

	Name	Data type	Description
@	crs_name	varchar(20)	

Uses

Name
dbo.Delete_Course
dbo.Course

Script

```
/* ------------------------------------------------------------------------------- */
/*                              Delete Course                                      */
/* ------------------------------------------------------------------------------- */

create   procedure Delete_Course @crs_name varchar(20)
as
if exists (select crs_name from [Course] where crs_name = @crs_name)
           delete from [Course] where crs_name = @crs_name
else
           select 'There is no course named ' + @crs_name
```

2.3.5. Procedure: dbo.Delete_Department

Input/Output

	Name	Data type	Description
→@	dept_name	varchar(20)	

Uses

Name
dbo.Delete_Department
dbo.Department

Script

```
/* ------------------------------------------------------------------------- */
/*                            Delete Department                            */
/* ------------------------------------------------------------------------- */

create   procedure Delete_Department @dept_name varchar(20)
as
if exists (select dept_name from [Department] where dept_name = @dept_name)
begin
           begin try
                      delete from [Department] where dept_name = @dept_name
                      return  1 -- deleted successfully
           end try
           begin catch
                      select 'Please check for instructors and students in this department'
                      return 0
           end catch
end
else
begin
           select 'No department with name ' + @dept_name
           return 0
end
```

2.3.6. Procedure: dbo.Delete_Topic

Input/Output

	Name	Data type	Description
→@	top_name	varchar(20)	

Uses

Name
dbo.Delete_Topic
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------- */
/*                                  Delete Topic                                 */
/* ------------------------------------------------------------------------------- */

create   procedure Delete_Topic @top_name varchar(20)
as
if exists (select top_name from [Topic] where top_name = @top_name)
           begin
           begin try
                                 delete from Topic where top_name = @top_name
           end try
           begin catch
                                 select 'Error'
           end catch
           end
else
           select 'There is no topic named ' + @top_name
```

2.3.7. Procedure: dbo.deleteExam

Input/Output

	Name	Data type	Description
@	ex_id	int	

Uses

Name
dbo.deleteExam
dbo.Exam
dbo.Exam_Answer
dbo.Exam_Question

Script

```
-- TODO : Handle Student/Course enrollement

/* ------------------------------------------------------------------------------- */
/*                                      Delete Exam                               */
/* ------------------------------------------------------------------------------- */

CREATE   PROC deleteExam @ex_id int
AS
BEGIN
           IF NOT EXISTS(select ex_id from Exam where ex_id = @ex_id)
                      SELECT 'Exam not found'
           ELSE
                      BEGIN
                                 BEGIN TRY
                                 BEGIN TRANSACTION -- Fathy Comment: Should we adjust other update procedures to include
transaction as well? Because If update fails, identity values get messed up
                                            -- Get the corresponding student and course and delete the grades of that student
                                            DECLARE @std_id int, @crs_id int
                                            SELECT @std_id = std_id from Exam_Answer WHERE ex_id = @ex_id
                                            SELECT @crs_id = crs_id from Exam WHERE ex_id = @ex_id

                                            -- Delete the Exam Answers
                                            DELETE FROM Exam_Answer
                                            WHERE ex_id = @ex_id

                                            -- Delete the Exam Questions
                                            DELETE FROM Exam_Question
                                            WHERE ex_id = @ex_id

                                            -- Delete the Exam itself
                                            DELETE FROM Exam
                                            WHERE ex_id = @ex_id
                                 COMMIT
                                 END TRY
                                 BEGIN CATCH
                                            SELECT 'Failed to delete the exam'
                                            ROLLBACK;
                                 END CATCH
                      END
END
```

2.3.8. Procedure: dbo.deleteInstructor

Input/Output

	Name	Data type	Description
→@	ins_id	int	

Uses

Name
dbo.deleteInstructor
dbo.Instructor
dbo.User

Script

```
/* ---------------------------------------------------------------------------- */
/*                            Delete Instructor                                */
/* ---------------------------------------------------------------------------- */

CREATE   PROCEDURE deleteInstructor
    @ins_id INTEGER
AS
BEGIN
    BEGIN TRY
    -- FIXME delete course attendance
    -- FIXME handle Ins_Course
    -- FIXME handle if instructor is a manager of a department


    DELETE FROM [Instructor]
    WHERE ins_id = @ins_id;

    DELETE FROM [User]
    WHERE usr_id = @ins_id;
    END TRY
    BEGIN CATCH
        SELECT 'failed to delete instructor' as [Error Message];
    END CATCH
END
```

2.3.9. Procedure: dbo.deleteQuestion

Input/Output

	Name	Data type	Description
@	q_id	int	

Uses

Name
dbo.deleteQuestion
dbo.MCQ
dbo.Question

Script

```
/* ------------------------------------------------------------------------------ */
/*                             Delete Question                                  */
/* ------------------------------------------------------------------------------ */

CREATE   PROC deleteQuestion @q_id int
AS
BEGIN
           IF EXISTS (select q_id from MCQ where q_id = @q_id)
           BEGIN
                      BEGIN TRY
                                 DELETE FROM MCQ
                                 WHERE q_id = @q_id
                                 ---------------------
                                 DELETE FROM Question
                                 WHERE q_id = @q_id
                      END TRY
                      BEGIN CATCH
                                 select 'This MCQ has been answered in an exam before'
                      END CATCH
           END
           ELSE
           BEGIN
                      BEGIN TRY
                                 DELETE FROM Question
                                 WHERE q_id = @q_id
                      END TRY
                      BEGIN CATCH
                                 select 'This TFQ has been answered in an exam before'
                      END CATCH
           END
END
```

2.3.10. Procedure: dbo.deleteStudent

Input/Output

	Name	Data type	Description
→@	std_id	int	

Uses

Name
dbo.deleteStudent
dbo.Student
dbo.User

Script

```
/* ---------------------------------------------------------------------------- */
/*                              Delete Student                                  */
/* ---------------------------------------------------------------------------- */

CREATE   PROCEDURE deleteStudent
    @std_id INTEGER
AS
BEGIN
    BEGIN TRY
    -- FIXME delete course attendance
    -- FIXME delete exam answers

    DELETE FROM [Student]
    WHERE std_id = @std_id;

    DELETE FROM [User]
    WHERE usr_id = @std_id;
    END TRY
    BEGIN CATCH
        SELECT 'failed to delete student' as [Error Message];
    END CATCH
END
```

2.3.11. Procedure: dbo.End_Course_for_Student

Input/Output

	Name	Data type	Description
→@	crs_name	varchar(20)	
→@	std_id	int	

Uses

Name
dbo.End_Course_for_Student
dbo.course
dbo.Course_Attendance
dbo.Student

Script

```
/* ------------------------------------------------------------------------------- */
/*                             End Course for Student                              */
/* ------------------------------------------------------------------------------- */

create   procedure End_Course_for_Student @crs_name varchar(20), @std_id int
as
if exists (select crs_name from [course] where crs_name = @crs_name)
           begin
                      if exists (select std_id from [Student] where std_id = @std_id)
                                 begin
                                            declare @id_course int
                                            select @id_course = crs_id from [Course] where crs_name = @crs_name
                                                       if exists (select crs_id, std_id from [Course_Attendance]
                                                                                         where (crs_id = @id_course and std_id
= @std_id))
                                                                  begin
                                                                             delete from [Course_Attendance]
                                                                             where (crs_id = @id_course and std_id = @std_id)
                                                                  end
                                                       else
                                                                  select 'This student does not take this course'
                                 end
                      else
                                 select 'There is no Course with this ID'
           end
else
           select 'There is no Course named ' + @crs_name
```

2.3.12. Procedure: dbo.End_Course_with_Instructor

Input/Output

	Name	Data type	Description
→@	crs_name	varchar(20)	
→@	ins_id	int	

Uses

Name
dbo.End_Course_with_Instructor
dbo.course
dbo.Ins_Course
dbo.Instructor

Script

```
/* ------------------------------------------------------------------------------- */
/*                          End Course with Instructor                             */
/* ------------------------------------------------------------------------------- */

create   procedure End_Course_with_Instructor @crs_name varchar(20), @ins_id int
as
if not exists (select crs_name from [course] where crs_name = @crs_name)
           begin
                      if not exists (select ins_id from [Instructor] where ins_id = @ins_id)
                                 begin
                                            delete from [Ins_Course] where
                                            (ins_id = @ins_id and
                                            crs_id = (select crs_id from [Course] where crs_name = @crs_name))
                                 end
                      else
                                 select 'There is no Instructor with this ID'
           end
else
           select 'There is no Course named ' + @crs_name
```

2.3.13. Procedure: dbo.generateExam

Input/Output

	Name	Data type	Description
→@	crs_name	varchar(100)	
→@	std_id	int	
→@→	ex_id	int	

Uses

Name
dbo.generateExam
dbo.Course
dbo.Course_Attendance
dbo.Exam
dbo.Exam_Answer
dbo.Exam_Question
dbo.Question
dbo.Student
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------- */
/*                                Exam table CRUDs                                */
/* ------------------------------------------------------------------------------- */

/* ------------------------------------------------------------------------------- */
/*                         Generate Exam for a specific course                     */
/* ------------------------------------------------------------------------------- */

CREATE   PROC generateExam @crs_name varchar(100), @std_id int, @ex_id int output
AS
BEGIN
           IF NOT EXISTS(SELECT crs_name FROM Course WHERE crs_name = @crs_name) OR NOT EXISTS (Select std_id from Student
WHERE std_id = @std_id)
                      SELECT 'Course or Student not found'
           ELSE
                      BEGIN
                      -- Get course ID
                                 DECLARE @crs_id int;
                                 SELECT @crs_id = crs_id FROM Course Where crs_name = @crs_name
                      IF NOT EXISTS (Select std_id from Course_Attendance WHERE std_id = @std_id AND crs_id = @crs_id)
                                 SELECT 'Student not enrolled in this course'
                      ELSE
                                 BEGIN
                                            -- Create exam instance and get the exam ID
                                            INSERT INTO Exam(date, crs_id)
                                                       VALUES(GETDATE(), @crs_id)
                                            SELECT @ex_id = SCOPE_IDENTITY()

                                            -- Create Cursor for row by row insertion in other tables
                                            DECLARE C1 Cursor
                                            -- Statement will return 10 random questions IDs for specified course
                                            -- with this assumption in mind ( 3 TF & 7 MCQ )
                                            FOR SELECT *
                                                                  FROM (SELECT top(3)q.q_id
                                                                             FROM Question q, Topic t, Course c
                                                                             WHERE q_type ='TF'
                                                                                        AND q.top_id = t.top_id
                                                                                        AND c.crs_id = t.crs_id
                                                                                        AND c.crs_name = @crs_name
                                                                             ORDER BY NEWID()) TF
                                                                  UNION ALL
                                                                  SELECT *
                                                                  FROM (
                                                                             SELECT top(7)q.q_id
                                                                             FROM Question q, Topic t, Course c
                                                                             WHERE q_type ='MCQ'
                                                                                        AND q.top_id = t.top_id
                                                                                        AND c.crs_id = t.crs_id
                                                                                        AND c.crs_name = @crs_name
                                                                             ORDER BY NEWID()) M
                                            FOR read only
                                            DECLARE @q_id int
                                            OPEN C1
                                            FETCH C1 INTO @q_id

                                            WHILE @@FETCH_STATUS = 0
                                            BEGIN
                                                       -- INSERT the q_id in tables Exam_Answer & Exam_Question
                                                       INSERT INTO Exam_Question (ex_id, q_id)
                                                                  VALUES (@ex_id, @q_id)
                                                                  -- NOTE: @ex_id is a fixed value and doesn't change with
the cursor


                                                       INSERT INTO Exam_Answer( std_id, ex_id, q_id)
                                                                  VALUES(@std_id, @ex_id, @q_id)
                                                                  -- NOTE: @ex_id and @std_id are fixed values and don't
change with the cursor


                                                       FETCH C1 INTO @q_id
                                            END
                                            CLOSE C1
                                            DEALLOCATE C1
                                 END

                      END
END
```

2.3.14. Procedure: dbo.GET_QUESTIONS_for_STUDENT_EXAM

Input/Output

	Name	Data type	Description
→@	exam_id	int	
→@	stduent_id	int	

Uses

Name
dbo.GET_QUESTIONS_for_STUDENT_EXAM
dbo.Exam
dbo.Exam_Answer
dbo.Exam_Question
dbo.Question
dbo.Student

Script

```
/*      --------------------------------------------------------------------------------     */
/*                    Report that takes exam number and the student ID then
/*                                returns the Questions in this exam with the student answers.
*/
/*     --------------------------------------------------------------------------------      */

create   procedure GET_QUESTIONS_for_STUDENT_EXAM @exam_id int, @stduent_id int
as
if exists (select ex_id from [Exam] where ex_id = @exam_id)
           begin
                      if exists (select std_id from Student where std_id = @stduent_id)
                                 begin
                                            select q.q_text, q.q_type, ea.std_answer, q.corr_answer
                                            from Exam_Answer ea
                                            inner join Exam_Question eq
                                            on ea.ex_id = eq.ex_id
                                            inner join Question q
                                            on eq.q_id = q.q_id
                                            where (ea.ex_id = @exam_id and ea.std_id = @stduent_id)
                                 end
                      else
                                 select CONCAT('There is no student with this ID', @stduent_id)
           end
else
           select CONCAT('There is no exam with this ID', @exam_id)
```

2.3.15. Procedure: dbo.Get_Questions_in_Exam

Input/Output

	Name	Data type	Description
@	ex_id	int	

Uses

Name
dbo.Get_Questions_in_Exam
dbo.Exam
dbo.Exam_Question
dbo.MCQ
dbo.Question

Script

```
/* ------------------------------------------------------------------------------ */
/*    Report that takes exam number and returns the Questions in it and choices           */
/* ------------------------------------------------------------------------------ */

-- Fathy Comment: It is mentioned it should be a "Freeform report", refer to this link
-- https://docs.microsoft.com/en-us/sql/reporting-services/tutorial-creating-a-free-form-report-report-builder?view=sql-
server-ver15

create   procedure Get_Questions_in_Exam @ex_id int
as
if exists(select ex_id from Exam where ex_id = @ex_id)
begin
           select q.q_text, q.q_type, mcq.ch_a, mcq.ch_b, mcq.ch_c, mcq.ch_d
           from Exam e
           inner join Exam_Question eq
           on e.ex_id = eq.ex_id
           inner join Question q
           on eq.q_id = q.q_id
           left join MCQ mcq
           on q.q_id = mcq.q_id
           where e.ex_id = @ex_id
end
else
           select 'Wrong Exam ID'
```

2.3.16. Procedure: dbo.getAllInstructors

Script

```
/* -------------------------------------------------------------------------- */
/*                             Read Instructor                               */
/* -------------------------------------------------------------------------- */

CREATE   PROCEDURE getAllInstructors
AS
BEGIN
    SELECT *
    FROM v_Instructors;
END
```

2.3.17. Procedure: dbo.getAllStudents

Uses

Name
dbo.getAllStudents
dbo.v_Students

Script

```
/* ------------------------------------------------------------------------- */
/*                              Read Student                                */
/* ------------------------------------------------------------------------- */


CREATE   PROCEDURE getAllStudents
AS
BEGIN
    SELECT *
    FROM v_Students;
END
```

2.3.18. Procedure: dbo.getDepartment

Input/Output

	Name	Data type	Description
→@	dept_id	int	

Uses

Name
dbo.getDepartment
dbo.Department
dbo.User

Script

```
/* ------------------------------------------------------------------------- */
/*                            Read Department                                */
/* ------------------------------------------------------------------------- */

CREATE   PROCEDURE getDepartment
    @dept_id INT
AS
BEGIN
    IF EXISTS (SELECT dept_id FROM Department WHERE dept_id = @dept_id)
        BEGIN
            SELECT D.dept_name, D.mgr_id, U.f_name + ' ' + U.l_name AS [Manager Name]
            FROM Department D, [User] U
            WHERE D.mgr_id = U.usr_id
        END
    ELSE
        SELECT 'Department ID does not exist' AS [Error Message]
END
```

2.3.19. Procedure: dbo.getDeptData

Input/Output

	Name	Data type	Description
→@	dept_id	int	

Uses

Name
dbo.getDeptData
dbo.Department
dbo.User

Script

```
/* ------------------------------------------------------------------------- */
/*                            Read Department                               */
/* ------------------------------------------------------------------------- */

-- Fathy Comment: This procedure was missing

CREATE   PROCEDURE getDeptData
    @dept_id INT
AS
BEGIN
    IF EXISTS (SELECT dept_id FROM Department WHERE dept_id = @dept_id)
        BEGIN
            SELECT D.dept_name, D.mgr_id, U.f_name + ' ' + U.l_name AS [Manager Name]
            FROM Department D, [User] U
            WHERE D.mgr_id = U.usr_id
        END
    ELSE
        SELECT 'Department ID does not exist' AS [Error Message]
END
```

2.3.20. Procedure: dbo.getInstructorsInDepartment

Input/Output

	Name	Data type	Description
→@	dept_id	int	

Script

```
CREATE   PROCEDURE getInstructorsInDepartment
    @dept_id INTEGER
AS
BEGIN
    SELECT usr_id, f_name,
        l_name,
        address,
        email,
        salary,
        degree
    FROM v_Instructors
    WHERE dept_id = @dept_id;
END
```

2.3.21. Procedure: dbo.getStudentAnswer

Input/Output

	Name	Data type	Description
→@	exam_id	int	
→@	stduent_id	int	

Uses

Name
dbo.getStudentAnswer
dbo.Exam
dbo.Exam_Answer
dbo.Exam_Question
dbo.Question
dbo.Student

Script

```
/*      ---------------------------------------------------------------------------------     */
/*                    Report that takes exam number and the student ID then
/*                               returns the Questions in this exam with the student answers.
*/
/*     ---------------------------------------------------------------------------------      */

create   procedure getStudentAnswer @exam_id int, @stduent_id int
as
if exists (select ex_id from [Exam] where ex_id = @exam_id)
           begin
                      if exists (select std_id from Student where std_id = @stduent_id)
                                 begin
                                            select q.q_text, q.q_type, ea.std_answer, q.corr_answer
                                            from Exam_Answer ea
                                            inner join Exam_Question eq
                                            on ea.ex_id = eq.ex_id
                                            inner join Question q
                                            on eq.q_id = q.q_id
                                            where (ea.ex_id = @exam_id and ea.std_id = @stduent_id)
                                 end
                      else
                                 select CONCAT('There is no student with this ID', @stduent_id)
           end
else
           select CONCAT('There is no exam with this ID', @exam_id)
```

2.3.22. Procedure: dbo.getStudentsInDepartment

Input/Output

	Name	Data type	Description
→@	dept_id	int	

Uses

Name
dbo.getStudentsInDepartment
dbo.v_Students

Script

```
/*      ---------------------------------------------------------------------------------     */
/*    Report that returns the students information according to Department No parameter     */
/*     ---------------------------------------------------------------------------------      */

CREATE   PROCEDURE getStudentsInDepartment
    @dept_id INTEGER
AS
BEGIN
    SELECT usr_id, f_name,
        l_name,
        address,
        email
    FROM v_Students
    WHERE dept_id = @dept_id;
END
```

2.3.23. Procedure: dbo.Insert_Course

Input/Output

	Name	Data type	Description
	crs_name	varchar(20)	

Uses

Name
dbo.Insert_Course
dbo.Course

Script

```
/* -------------------------------------------------------------------------------- */
/*                                   Course CRUDs                                  */
/* -------------------------------------------------------------------------------- */


/* -------------------------------------------------------------------------------- */
/*                                Insert Course                                    */
/* -------------------------------------------------------------------------------- */

create   procedure Insert_Course @crs_name varchar(20)
as
if not exists (select @crs_name from [Course] where crs_name = @crs_name)
           insert into [course] values(@crs_name)
else
           select 'This course already exists' as [Error Message]
```

2.3.24. Procedure: dbo.Insert_Department

Input/Output

	Name	Data type	Description
→@	dept_name	varchar(20)	
→@	id_mgr	int	
↔@	dept_id	int	

Uses

Name
dbo.Insert_Department
dbo.Department
dbo.Instructor

Used By

Name
dbo.Insert_Department
dbo.Insert_Department_With_Manager

Script

```
/* ----------------------- required helping procedure ----------------------- */

/* -------------------------------------------------------------------------- */
/*                          Create Department                               */
/* -------------------------------------------------------------------------- */

-- Department [dept_id, dept_name, mgr_id]
create   procedure Insert_Department @dept_name varchar(20), @id_mgr int, @dept_id int output
as
           if exists (select ins_id from [Instructor] where ins_id = @id_mgr)
           begin
                      insert into [Department] values (@dept_name, @id_mgr)
                      select @dept_id = dept_id from Department where dept_name = @dept_name
                      return 1
           end
           else
           begin
                      select 'no instructor with ID ' + cast(@id_mgr as varchar) +' found'
                      return 0
           end
-------------------------------------------------------------------------
```

2.3.25. Procedure: dbo.Insert_Department_With_Manager

Input/Output

	Name	Data type	Description
→@	dept_name	varchar(100)	
→@	f_name	varchar(50)	
→@	l_name	varchar(50)	
→@	address	varchar(150)	
→@	email	varchar(90)	
→@	password	varchar(255)	
→@	salary	money	
→@	degree	varchar(50)	
→@→	dept_id	int	
→@→	mgr_id	int	

Uses

Name
dbo.Insert_Department_With_Manager
dbo.Instructor
dbo.Insert_Department
dbo.Insert_Instructor

Script

```
/* ------------------------------------------------------------------------- */
/*                   Create Department with manager                          */
/* ------------------------------------------------------------------------- */

CREATE   PROCEDURE [dbo].[Insert_Department_With_Manager]
    @dept_name varChar(100),
    @f_name varChar(50),
    @l_name varChar(50),
    @address varChar(150),
    @email varChar(90),
    @password varChar(255),
    @salary MONEY,
    @degree varChar(50),
    @dept_id INTEGER OUTPUT,
    @mgr_id INTEGER OUTPUT
AS
BEGIN
    -- TODO use try catch for errors
    ALTER TABLE Instructor NOCHECK CONSTRAINT Instructor_fk_1;
    DECLARE @_no_dep INT = 0;
    Exec [dbo].[Insert_Instructor] @f_name, @l_name, @address, @email, @password, @salary, @degree, @_no_dep, @mgr_id OUTPUT;
    -- TODO replace this with the real procedure
    Exec [dbo].[Insert_Department] @dept_name, @mgr_id, @dept_id OUTPUT;
    UPDATE Instructor SET dept_id = @dept_id WHERE ins_id = @mgr_id;
    ALTER TABLE Instructor CHECK CONSTRAINT Instructor_fk_1;

END
```

2.3.26. Procedure: dbo.Insert_Instructor

Input/Output

	Name	Data type	Description
→@	f_name	varchar(50)	
→@	l_name	varchar(50)	
→@	address	varchar(150)	
→@	email	varchar(90)	
→@	password	varchar(255)	
→@	salary	money	
→@	degree	varchar(50)	
→@	dept_id	int	
→@→	ins_id	int	

Uses

Name
dbo.Insert_Instructor
dbo.Instructor
PRIVATE.Insert_User

Used By

Name
dbo.Insert_Instructor
dbo.Insert_Department_With_Manager

Script

```
/* -------------------------------------------------------------------------- */
/*                          Create Instructor                                 */
/* -------------------------------------------------------------------------- */

-- Instructor [ins_id, salary, degree, dept_id]
/*
user has type 'I' capital I
*/
CREATE   PROCEDURE [dbo].[Insert_Instructor]
    @f_name varChar(50),
    @l_name varChar(50),
    @address varChar(150),
    @email varChar(90),
    @password varChar(255),
    @salary MONEY,
    @degree varChar(50),
    @dept_id INTEGER,
    @ins_id INTEGER OUTPUT
AS
BEGIN
    BEGIN TRY
    DECLARE @usr_id INTEGER;
    Exec [PRIVATE].[Insert_User] 'I', @f_name, @l_name, @address, @email, @password, @usr_id OUTPUT;
    INSERT INTO [Instructor]
        (ins_id, salary, degree, dept_id)
    VALUES
        (
            @usr_id,
            @salary,
            @degree,
            @dept_id
    );
    SET @ins_id = @usr_id;
    END TRY
    BEGIN CATCH
        SELECT 'failed to insert instructor' as [Error Message];
    END CATCH
END
```

2.3.27. Procedure: dbo.Insert_Student

Input/Output

	Name	Data type	Description
→@	f_name	varchar(50)	
→@	l_name	varchar(50)	
→@	address	varchar(150)	
→@	email	varchar(90)	
→@	password	varchar(255)	
→@	dept_id	int	
→@→	stu_id	int	

Uses

Name
dbo.Insert_Student
dbo.Student
PRIVATE.Insert_User

Script

```
/* -------------------------------------------------------------------------- */
/*                            Create Student                                  */
/* -------------------------------------------------------------------------- */

-- Student [std_id, dept_id]
/*
user has type 'S' capital S
*/
CREATE   PROCEDURE [dbo].[Insert_Student]
    @f_name varChar(50),
    @l_name varChar(50),
    @address varChar(150),
    @email varChar(90),
    @password varChar(255),
    @dept_id INTEGER,
    @stu_id INTEGER OUTPUT
AS
BEGIN
    begin try
    Exec [PRIVATE].[Insert_User] 'S', @f_name, @l_name, @address, @email, @password, @stu_id OUTPUT;
    INSERT INTO [Student]
    VALUES
        (
            @stu_id,
            @dept_id
    );
    END TRY
    BEGIN CATCH
        SELECT 'failed to insert student' as [Error Message];
    END CATCH
END
```

2.3.28. Procedure: dbo.Insert_Topic

Input/Output

	Name	Data type	Description
→@	top_name	varchar(20)	
→@	crs_name	varchar(20)	

Uses

Name
dbo.Insert_Topic
dbo.Course
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------- */
/*                                Topic CRUDs                                     */
/* ------------------------------------------------------------------------------- */

/* ------------------------------------------------------------------------------- */
/*                                Insert Topic                                    */
/* ------------------------------------------------------------------------------- */

create   procedure Insert_Topic @top_name varchar(20), @crs_name varchar(20)
as
declare @id_crs int
if not exists (select top_name from [Topic] where top_name = @top_name)
           begin
                      begin try
                                 select @id_crs = crs_id from [Course] where crs_name = @crs_name
                                 insert into Topic values (@top_name, @id_crs)
                      end try
                      begin catch
                                 select 'There is no course named ' + @crs_name
                      end catch
           end
else
           select 'This Topic already exists'
```

2.3.29. Procedure: dbo.insertMCQ

Input/Output

	Name	Data type	Description
→@	top_id	int	
→@	q_text	varchar(300)	
→@	ch_a	varchar(300)	
→@	ch_b	varchar(300)	
→@	ch_c	varchar(300)	
→@	ch_d	varchar(300)	
→@	corr_answer	varchar(1)	
→@→	q_id	int	

Uses

Name
dbo.insertMCQ
dbo.MCQ
dbo.Topic
PRIVATE.insertQuestion

Script

```
/* ------------------------------------------------------------------------------ */
/*                              MCQ Question                                      */
/* ------------------------------------------------------------------------------ */

-- MCQ [q_id, ch_a, ch_b, ch_c, ch_d]
CREATE   PROC insertMCQ
                      @top_id int,
                      @q_text varchar(300),
                      @ch_a varchar(300),
                      @ch_b varchar(300),
                      @ch_c varchar(300),
                      @ch_d varchar(300),
                      @corr_answer varchar(1),
                      @q_id int output
AS
BEGIN
           IF NOT EXISTS( SELECT top_id FROM Topic WHERE top_id = @top_id)
                      SELECT 'Make sure topic already exists'
           ELSE
                      BEGIN
                                 BEGIN TRY
                                            EXECUTE [PRIVATE].insertQuestion @top_id, 'MCQ', @q_text, @corr_answer, @q_id
output
                                            INSERT INTO MCQ (q_id, ch_a, ch_b, ch_c, ch_d)
                                                       VALUES(@q_id, @ch_a, @ch_b, @ch_c, @ch_d)
                                 END TRY
                                 BEGIN CATCH
                                            select 'Make sure you entered the data correctly'
                                 END CATCH
                      END
END
```

2.3.30. Procedure: dbo.insertTFQ

Input/Output

	Name	Data type	Description
→@	top_id	int	
→@	q_text	varchar(300)	
→@	corr_answer	varchar(1)	
→@→	q_id	int	

Uses

Name
dbo.insertTFQ
dbo.Topic
PRIVATE.insertQuestion

Script

```
/* ------------------------------------------------------------------------------- */
/*                        True or False Question                                 */
/* ------------------------------------------------------------------------------- */

CREATE   PROC insertTFQ
                      @top_id int,
                      @q_text varchar(300),
                      @corr_answer varchar(1),
                      @q_id int output
AS
BEGIN
           IF NOT EXISTS( SELECT top_id FROM Topic WHERE top_id = @top_id)
                      SELECT 'Make sure topic already exists'
           ELSE
                      BEGIN
                                 BEGIN TRY
                                            EXECUTE [PRIVATE].insertQuestion @top_id, 'TF', @q_text, @corr_answer, @q_id
output
                                 END TRY
                                 BEGIN CATCH
                                            SELECT 'Make sure data is correct'
                                 END CATCH
                      END
END
```

2.3.31. Procedure: dbo.returnGrades

Input/Output

	Name	Data type	Description
→@	std_id	int	

Uses

Name
dbo.returnGrades
dbo.Course
dbo.Course_Attendance
dbo.Student

Script

```
CREATE   PROCEDURE dbo.returnGrades
@std_id INT

AS
BEGIN
           DECLARE @t TABLE
           (
                      crs_id INT,
                      crs_name VARCHAR(100),
                      std_grade INT
           )
    IF EXISTS (SELECT std_id FROM Student WHERE std_id = @std_id)
                      BEGIN
                                 INSERT INTO @t (crs_id, crs_name, std_grade)
                                 SELECT CA.crs_id, C.crs_name, CA.grade
                                 FROM Course_Attendance CA, Course C
                                 WHERE CA.crs_id = C.crs_id AND CA.std_id = @std_id
                                 SELECT * FROM @t
                      END
    ELSE
        SELECT 'Student ID does not exist' AS [Error Message]
END
```

2.3.32. Procedure: dbo.setCourseName

Input/Output

	Name	Data type	Description
@	crs_id	int	
@	crs_name	varchar(50)	

Uses

Name
dbo.setCourseName
dbo.Course

Script

```
/* ------------------------------------------------------------------------------- */
/*                         Update Course Name                                     */
/* ------------------------------------------------------------------------------- */

CREATE   PROCEDURE setCourseName @crs_id INT, @crs_name VARCHAR(50)
AS
BEGIN
    IF EXISTS(SELECT crs_id FROM Course WHERE crs_id = @crs_id)
        BEGIN
            UPDATE [Course]
            SET crs_name = @crs_name
            WHERE crs_id = @crs_id
        END

    ELSE
        SELECT 'Course ID not found' AS [Error Message]
END
```

2.3.33. Procedure: dbo.setTopicName

Input/Output

	Name	Data type	Description
→@	top_id	int	
→@	top_name	varchar(50)	

Uses

Name
dbo.setTopicName
dbo.Topic

Script

```
/* -------------------------------------------------------------------------------- */
/*                               Update Topic Name                                */
/* -------------------------------------------------------------------------------- */


CREATE   PROCEDURE setTopicName @top_id INT, @top_name VARCHAR(50)
AS
BEGIN
    IF EXISTS(SELECT top_id FROM Topic WHERE top_id = @top_id)
        BEGIN
            UPDATE [Topic]
            SET top_name = @top_name
            WHERE top_id = @top_id
        END

    ELSE
        SELECT 'Topic ID not found' AS [Error Message]
END
```

2.3.34. Procedure: dbo.sp_returngrades

Input/Output

	Name	Data type	Description
@	std_id	int	

Uses

Name
dbo.sp_returngrades
dbo.Course
dbo.Course_Attendance
dbo.Student

Script

```
CREATE   PROCEDURE dbo.sp_returngrades
@std_id INT

AS
BEGIN
    IF EXISTS (SELECT std_id FROM Student WHERE std_id = @std_id)
                      BEGIN
                                 SELECT CA.crs_id, C.crs_name, CA.grade
                                 FROM Course_Attendance CA, Course C
                                 WHERE CA.crs_id = C.crs_id AND CA.std_id = @std_id
                      END
    ELSE
        SELECT 'Student ID does not exist' AS [Error Message]
END
```

2.3.35. Procedure: dbo.Student_Take_course_with_Instructor

Input/Output

	Name	Data type	Description
→@	std_id	int	
→@	crs_id	int	
→@	ins_id	int	

Uses

Name
dbo.Student_Take_course_with_Instructor
dbo.Course
dbo.Course_Attendance
dbo.Ins_Course
dbo.Instructor
dbo.Student

Script

```
/* ------------------------------------------------------------------------------ */
/*            Student, Course, Instructor CRUDs (Course_Attendance table)          */
/* ------------------------------------------------------------------------------ */

/* ------------------------------------------------------------------------------ */
/*                      Student Take Course with Instructor                       */
/* ------------------------------------------------------------------------------ */

create   procedure Student_Take_course_with_Instructor @std_id int, @crs_id int, @ins_id int
as
BEGIN TRY
if exists (select ins_id from [Instructor] where ins_id = @ins_id)
and exists (select std_id from [Student] where std_id = @std_id)
           begin
                      if exists (select crs_id from [Course] where crs_id = @crs_id)
                                 begin
                                            if exists (select crs_id, ins_id from [Ins_Course]
                                            where (crs_id = @crs_id and ins_id = @ins_id))
                                            insert into Course_Attendance (crs_id, std_id, ins_id)
                                            values(@crs_id, @std_id, @ins_id)

                                            else
                                                       select 'This course is not assgined to this instructor'
                                 end
                      else
                                 select 'There is no course with this ID'
           end

else
           select 'Please check the Instructor and Student ID'
END TRY
BEGIN CATCH
           SELECT 'Duplicate data , please check your data' AS [Error Message]
END CATCH
```

2.3.36. Procedure: dbo.Topics_of_Course

Input/Output

	Name	Data type	Description
→@	crs_name	varchar(20)	

Uses

Name
dbo.Topics_of_Course
dbo.Course
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------- */
/*                              Report that takes course ID and returns its topics              */
/* ------------------------------------------------------------------------------- */

create   procedure Topics_of_Course @crs_name varchar(20)
as
if exists(select crs_name from Course where crs_name = @crs_name)
begin
           select t.top_name
           from Course c
           inner join Topic t
           on c.crs_id = t.crs_id
           where c.crs_name = @crs_name
end
else
           select 'There is no course named ' + @crs_name
```

2.3.37. Procedure: dbo.Update_Department_Manager

Input/Output

	Name	Data type	Description
@	dept_name	varchar(20)	
@	mgr_id	int	

Uses

Name
dbo.Update_Department_Manager
dbo.Department

Script

```
/* -------------------------------------------------------------------------- */
/*                   Update Department Manager                               */
/* -------------------------------------------------------------------------- */

create   procedure Update_Department_Manager @dept_name varchar(20), @mgr_id int
as
if exists (select dept_name from [Department] where dept_name = @dept_name)
begin
           begin try
                      update [Department] set mgr_id = @mgr_id where dept_name = @dept_name
           end try
           begin catch
                      select 'Error: There is no an instructor with ID ' + @mgr_id
           end catch
end
else
           select 'No department with name ' + @dept_name
```

2.3.38. Procedure: dbo.updateInstructorData

Input/Output

	Name	Data type	Description
→@	f_name	varchar(50)	
→@	l_name	varchar(50)	
→@	address	varchar(150)	
→@	email	varchar(90)	
→@	salary	money	
→@	degree	varchar(50)	
→@	dept_id	int	
→@	ins_id	int	

Uses

Name
dbo.updateInstructorData
dbo.Instructor
dbo.updateUserData

Script

```
/* -------------------------------------------------------------------------- */
/*                      Update Instructor Data                               */
/* -------------------------------------------------------------------------- */

CREATE   PROCEDURE updateInstructorData
    @f_name varChar(50),
    @l_name varChar(50),
    @address varChar(150),
    @email varChar(90),
    @salary MONEY,
    @degree varChar(50),
    @dept_id INTEGER,
    @ins_id INTEGER
AS
BEGIN
    BEGIN TRY
    -- update the userInfo
    EXEC [dbo].[updateUserData]
    @usr_id = @ins_id,
    @f_name = @f_name,
    @l_name = @l_name,
    @address = @address,
    @email = @email;
    -- update instructor specificInfo
    UPDATE [Instructor]
    SET salary = @salary,
        degree = @degree,
        dept_id = @dept_id
    WHERE ins_id = @ins_id;
    END TRY
    BEGIN CATCH
        -- TODO send specific error message when department id is not in the database
        SELECT 'failed to update instructor' as [Error Message];
    END CATCH
END
```

2.3.39. Procedure: dbo.updateMCQ

Input/Output

	Name	Data type	Description
→@	q_id	int	
→@	top_id	int	
→@	q_text	varchar(300)	
→@	ch_a	varchar(300)	
→@	ch_b	varchar(300)	
→@	ch_c	varchar(300)	
→@	ch_d	varchar(300)	
→@	corr_answer	varchar(1)	

Uses

Name
dbo.updateMCQ
dbo.MCQ
dbo.Question
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------- */
/*                            Update Question                                      */
/* ------------------------------------------------------------------------------- */

/* ------------------------------------------------------------------------------- */
/*                               Update MCQ                                        */
/* ------------------------------------------------------------------------------- */

CREATE   PROC updateMCQ
                      @q_id int,
                      @top_id int,
                      @q_text varchar(300),
                      @ch_a varchar(300),
                      @ch_b varchar(300),
                      @ch_c varchar(300),
                      @ch_d varchar(300),
                      @corr_answer varchar(1)
AS
BEGIN
-- Check for question existence
           IF NOT EXISTS( SELECT q_id FROM Question where q_id = @q_id)
                      SELECT 'Question does not exist'
           ELSE
                      BEGIN
                                 IF NOT EXISTS( SELECT top_id FROM Topic WHERE top_id = @top_id)
                                            SELECT 'Make sure topic already exists'
                                 ELSE
                                            BEGIN
                                                       BEGIN TRY
                                                       BEGIN TRANSACTION
                                                                  UPDATE Question
                                                                  SET
                                                                             top_id = @top_id,
                                                                             q_text = @q_text,
                                                                             corr_answer = @corr_answer
                                                                  WHERE q_id = @q_id;
                                                                  -------------------
                                                                  UPDATE MCQ
                                                                  SET
                                                                             ch_a = @ch_a,
                                                                             ch_b = @ch_b,
                                                                             ch_c = @ch_c,
                                                                             ch_d = @ch_d
                                                                  WHERE q_id = @q_id
                                                       COMMIT
                                                       END TRY
                                                       BEGIN CATCH
                                                                  select 'Make sure you entered the data correctly'
                                                                  ROLLBACK;
                                                       END CATCH
                                            END
                      END
END
```

2.3.40. Procedure: dbo.updateStudentData

Input/Output

	Name	Data type	Description
→@	f_name	varchar(50)	
→@	l_name	varchar(50)	
→@	address	varchar(150)	
→@	email	varchar(90)	
→@	dept_id	int	
→@	std_id	int	

Uses

Name
dbo.updateStudentData
dbo.Student
dbo.updateUserData

Script

```
/* ------------------------------------------------------------------------- */
/*                           Update Student Data                            */
/* ------------------------------------------------------------------------- */

CREATE   PROCEDURE updateStudentData
    @f_name varChar(50),
    @l_name varChar(50),
    @address varChar(150),
    @email varChar(90),
    @dept_id INTEGER,
    @std_id INTEGER
AS
BEGIN
    BEGIN TRY
    -- update the userInfo
    EXEC [dbo].[updateUserData]
    @usr_id = @std_id,
    @f_name = @f_name,
    @l_name = @l_name,
    @address = @address,
    @email = @email;
    -- update student specificInfo
    UPDATE [Student]
    SET dept_id = @dept_id
    WHERE std_id = @std_id;
    END TRY
    BEGIN CATCH
        -- TODO send specific error message when department id is not in the database
        SELECT 'failed to update student' as [Error Message];
    END CATCH
END
```

2.3.41. Procedure: dbo.updateTFQ

Input/Output

	Name	Data type	Description
@	q_id	int	
@	top_id	int	
@	q_text	varchar(300)	
@	corr_answer	varchar(1)	

Uses

Name
dbo.updateTFQ
dbo.Question
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------ */
/*                            Update True/False                                 */
/* ------------------------------------------------------------------------------ */

CREATE   PROC updateTFQ
                      @q_id int,
                      @top_id int,
                      @q_text varchar(300),
                      @corr_answer varchar(1)
AS
BEGIN
IF NOT EXISTS(SELECT q_id FROM Question where q_id = @q_id)
           SELECT 'Question does not exist'
ELSE
BEGIN
                      IF NOT EXISTS( SELECT top_id FROM Topic WHERE top_id = @top_id)
                      SELECT 'Make sure topic already exists'
           ELSE
                      BEGIN
                                 BEGIN TRY
                                            UPDATE Question
                                            SET
                                                       top_id = @top_id,
                                                       q_text = @q_text,
                                                       corr_answer = @corr_answer
                                            WHERE q_id = @q_id;
                                 END TRY
                                 BEGIN CATCH
                                            select 'Make sure you entered the data correctly'
                                 END CATCH
                      END
END
END
```

2.3.42. Procedure: dbo.updateUserData

Input/Output

	Name	Data type	Description
→@	usr_id	int	
→@	f_name	varchar(50)	
→@	l_name	varchar(50)	
→@	address	varchar(150)	
→@	email	varchar(90)	

Uses

Name
dbo.updateUserData
dbo.User

Used By

Name
dbo.updateUserData
dbo.updateInstructorData
dbo.updateStudentData

Script

```
/* ---------------------------------------------------------------------------- */
/*                            update user data                                  */
/* ---------------------------------------------------------------------------- */

CREATE   PROCEDURE updateUserData
    @usr_id INTEGER,
    @f_name varChar(50),
    @l_name varChar(50),
    @address varChar(150),
    @email varChar(90)
AS
BEGIN
    BEGIN TRY
    UPDATE [User]
    SET
        f_name = @f_name,
        l_name = @l_name,
        address = @address,
        email = @email
    WHERE usr_id = @usr_id;
    END TRY
    BEGIN CATCH
        -- TODO send specific error message when email is already in database
        SELECT 'failed to update user' as [Error Message];
    END CATCH
END
```

2.3.43. Procedure: dbo.viewCourseMCQ

Input/Output

	Name	Data type	Description
→@	crs_name	varchar(100)	

Uses

Name
dbo.viewCourseMCQ
dbo.Course
dbo.MCQ
dbo.Question
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------ */
/*                Display MCQ Question for a certain Course                       */
/* ------------------------------------------------------------------------------ */

CREATE   PROC viewCourseMCQ @crs_name varchar(100)
AS
BEGIN
           IF NOT EXISTS(select crs_id from Course where crs_name = @crs_name)
                      SELECT 'Course not found'
           ELSE
           BEGIN
                      Select q.q_id AS QID,
                      c.crs_name AS [Course],
                      T.top_name AS [Topic],
                      q.q_text AS [Question],
                      m.ch_a AS [Choice a],
                      m.ch_b AS [Choice b],
                      m.ch_c AS [Choice c],
                      m.ch_d AS [Choice d],
                      q.corr_answer AS [Correct Answer]
                      from Question q, MCQ m, Topic t, Course c
                      where q.q_id = m.q_id and t.top_id = q.top_id and c.crs_id = t.crs_id and c.crs_name = @crs_name;
           END
END
```

2.3.44. Procedure: dbo.viewCourseTFQ

Input/Output

	Name	Data type	Description
	crs_name	varchar(100)	

Uses

Name
dbo.viewCourseTFQ
dbo.Course
dbo.Question
dbo.Topic

Script

```
/* -------------------------------------------------------------------------------- */
/*                Display True/False Question for a certain Course                 */
/* -------------------------------------------------------------------------------- */

CREATE   PROC viewCourseTFQ @crs_name varchar(100)
AS
BEGIN
           IF NOT EXISTS(select crs_id from Course where crs_name = @crs_name)
                      SELECT 'Course not found'
           ELSE
           BEGIN
                      Select q.q_id AS QID,
                      c.crs_name AS [Course],
                      T.top_name AS [Topic],
                      q.q_text AS [Question],
                      q.corr_answer AS [Correct Answer]
                      from Question q, Topic t, Course c
                      where t.top_id = q.top_id and c.crs_id = t.crs_id and c.crs_name = @crs_name and q.q_type = 'TF';
           END
END
```

2.3.45. Procedure: dbo.viewExamQuestions

Input/Output

	Name	Data type	Description
→@	ex_id	int	

Uses

Name
dbo.viewExamQuestions
dbo.Exam
dbo.Exam_Question
dbo.MCQ
dbo.Question

Script

```
/* ------------------------------------------------------------------------------- */
/*                         Display Exam without Answers                            */
/* ------------------------------------------------------------------------------- */

CREATE   PROC viewExamQuestions @ex_id int
AS
BEGIN
           IF NOT EXISTS(select ex_id from Exam where ex_id = @ex_id)
                      SELECT 'Exam not found'
           ELSE
                      BEGIN
                                 SELECT e.ex_id, q.q_id, q.q_text, q.q_type
                                 FROM Exam e, Question q, Exam_Question eq
                                 WHERE e.ex_id = eq.ex_id
                                            AND q.q_id = eq.q_id
                                            AND q.q_type='TF'
                                            AND e.ex_id = @ex_id

                                 SELECT e.ex_id, q.q_id, q.q_text, q.q_type, M.ch_a, M.ch_b, M.ch_c, M.ch_d
                                 FROM Exam e, Question q, Exam_Question eq, MCQ M
                                 WHERE e.ex_id = eq.ex_id
                                            AND q.q_id = eq.q_id
                                            AND M.q_id = q.q_id
                                            AND q.q_type='MCQ'
                                            AND e.ex_id = @ex_id
                      END
END
```

2.3.46. Procedure: dbo.viewMCQ

Uses

Name
dbo.viewMCQ
dbo.Course
dbo.MCQ
dbo.Question
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------- */
/*                Display MCQ with choices and correct answer                      */
/* ------------------------------------------------------------------------------- */

CREATE   PROC viewMCQ
AS
BEGIN
           Select q.q_id AS QID,
                      c.crs_name AS [Course],
                      T.top_name AS [Topic],
                      q.q_text AS [Question],
                      m.ch_a AS [Choice a],
                      m.ch_b AS [Choice b],
                      m.ch_c AS [Choice c],
                      m.ch_d AS [Choice d],
                      q.corr_answer AS [Correct Answer]
           from Question q, MCQ m, Topic t, Course c
           where q.q_id = m.q_id and t.top_id = q.top_id and c.crs_id = t.crs_id
END
```

2.3.47. Procedure: dbo.viewTFQ

Uses

Name
dbo.viewTFQ
dbo.Course
dbo.Question
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------- */
/*                Display True/False with choices and correct answer              */
/* ------------------------------------------------------------------------------- */

CREATE   PROC viewTFQ
AS
BEGIN
           Select q.q_id AS QID,
                      c.crs_name AS [Course],
                      T.top_name AS [Topic],
                      q.q_text AS [Question],
                      q.corr_answer AS [Correct Answer]
           From Question q, Topic t, Course c
           Where q_type = 'TF'
END
```

2.3.48. Procedure: dbo.viewTopicMCQ

Input/Output

	Name	Data type	Description
→@	top_name	varchar(200)	

Uses

Name
dbo.viewTopicMCQ
dbo.Course
dbo.MCQ
dbo.Question
dbo.Topic

Script

```
/* ------------------------------------------------------------------------------ */
/*                             Read Question                                    */
/* ------------------------------------------------------------------------------ */

/* ------------------------------------------------------------------------------ */
/*                  Display MCQ Question for a certain topic                     */
/* ------------------------------------------------------------------------------ */

-- Fathy Comment : Is this procedure a report? Should it instead be reading question data using question id?

CREATE   PROC viewTopicMCQ @top_name varchar(200)
AS
BEGIN
           IF NOT EXISTS(select top_id from Topic where top_name = @top_name)
                      SELECT 'Topic not found'
           ELSE
           BEGIN
                      Select q.q_id AS QID,
                      c.crs_name AS [Course],
                      T.top_name AS [Topic],
                      q.q_text AS [Question],
                      m.ch_a AS [Choice a],
                      m.ch_b AS [Choice b],
                      m.ch_c AS [Choice c],
                      m.ch_d AS [Choice d],
                      q.corr_answer AS [Correct Answer]
                      from Question q, MCQ m, Topic t, Course c
                      where q.q_id = m.q_id and t.top_id = q.top_id and c.crs_id = t.crs_id and t.top_name = @top_name
           END
END
```

2.3.49. Procedure: dbo.viewTopicTFQ

Input/Output

	Name	Data type	Description
	top_name	varchar(200)	

Uses

Name
dbo.viewTopicTFQ
dbo.Course
dbo.Question
dbo.Topic

Script

```
/* -------------------------------------------------------------------------------- */
/*                Display True/False Question for a certain topic                  */
/* -------------------------------------------------------------------------------- */

CREATE   PROC viewTopicTFQ @top_name varchar(200)
AS
BEGIN
           IF NOT EXISTS(select top_id from Topic where top_name = @top_name)
                      SELECT 'Topic not found'
           ELSE
           BEGIN
                      Select q.q_id AS QID,
                      c.crs_name AS [Course],
                      T.top_name AS [Topic],
                      q.q_text AS [Question],
                      q.corr_answer AS [Correct Answer]
                      from Question q, Topic t, Course c
                      where t.top_id = q.top_id and c.crs_id = t.crs_id and t.top_name = @top_name and q.q_type = 'TF';
           END
END
```

2.3.50. Procedure: PRIVATE.Insert_User

Input/Output

	Name	Data type	Description
	user_type	varchar(1)	
	f_name	varchar(50)	
	l_name	varchar(50)	
	address	varchar(150)	
	email	varchar(90)	
	password	varchar(255)	
	usr_id	int	

Uses

Name
PRIVATE.Insert_User
dbo.User

Used By

Name
PRIVATE.Insert_User
dbo.Insert_Instructor
dbo.Insert_Student

Script

```sql
/* -------------------------------------------------------------------------- */
/*                                Create User                                */
/* -------------------------------------------------------------------------- */

-- User [usr_id, user_type, f_name, l_name, address, email, password]

CREATE   PROCEDURE [PRIVATE].[Insert_User]
    @user_type varChar(1),
    @f_name varChar(50),
    @l_name varChar(50),
    @address varChar(150),
    @email varChar(90),
    @password varChar(255),
    @usr_id INTEGER OUTPUT
AS
BEGIN
    BEGIN TRY



    DECLARE @hashed_password varChar(255);
    -- TODO define the seed globally
    SELECT @hashed_password = HASHBYTES('SHA2_256', @password+'seed');
    INSERT INTO [User]
        (user_type, f_name, l_name, address, email, [hashed_password])
    VALUES
        (
            @user_type,
            @f_name,
            @l_name,
            @address,
            @email,
            @hashed_password
    );

    SELECT @usr_id = scope_identity();
    /* NOTE scope_identity() may give wrong result when queries run in parrallel
    ref:[1]:https://blog.sqlauthority.com/2009/03/24/sql-server-2008-scope_identity-bug-with-multi-processor-parallel-plan-
and-solution/
    [2]:https://stackoverflow.com/questions/42648/sql-server-best-way-to-get-identity-of-inserted-row
    */
    END TRY
    BEGIN CATCH

    -- select ERROR_MESSAGE() 'Error Message'
           -- , ERROR_NUMBER() 'Error Number'
           -- , ERROR_LINE () 'Error Line Number'
           -- , ERROR_SEVERITY () 'Error Severity Level'
           -- , ERROR_PROCEDURE() 'Error Procedure'
           -- , ERROR_STATE () 'Error State';
    IF (ERROR_NUMBER() = 2627)
        SELECT 'User already exists' as [Error Message];
    ELSE
        SELECT ERROR_NUMBER() 'Error Number', ERROR_MESSAGE() 'Error Message';
    END CATCH
END
```

2.3.51. Procedure: PRIVATE.insertQuestion

Input/Output

	Name	Data type	Description
→@	top_id	int	
→@	q_type	varchar(3)	
→@	q_text	varchar(300)	
→@	corr_answer	varchar(1)	
→@→	q_id	int	

Uses

Name
PRIVATE.insertQuestion
dbo.Question
dbo.Topic

Used By

Name
PRIVATE.insertQuestion
dbo.insertMCQ
dbo.insertTFQ

Script

```
/* ------------------------------------------------------------------------------ */
/*                       Question Table CRUDs                                     */
/* ------------------------------------------------------------------------------ */

/* ------------------------------------------------------------------------------ */
/*                            Insert Question                                     */
/* ------------------------------------------------------------------------------ */

CREATE   PROC [PRIVATE].insertQuestion
                      @top_id int,
                      @q_type varchar(3),
                      @q_text varchar(300),
                      @corr_answer varchar(1),
                      @q_id int output
AS
BEGIN
           IF NOT EXISTS( SELECT top_id FROM Topic WHERE top_id = @top_id)
                      SELECT 'Make sure topic already exists'
           ELSE
                      BEGIN
                                 BEGIN TRY
                                            INSERT INTO Question(q_type, q_text, corr_answer, top_id)
                                                       VALUES(@q_type, @q_text, @corr_answer, @top_id)
                                            SELECT @q_id = SCOPE_IDENTITY();
                                 END TRY
                                 BEGIN CATCH
                                            SELECT 'Make sure you entered the data correctly'
                                 END CATCH
                      END
END
```

2.4. Functions

2.4.1. Function: dbo.getQuestionMark

Input/Output

	Name	Data type	Description
	Returns	int	
	q_id	int	

2.4.2. Function: dbo.getStudentGrade

Input/Output

	Name	Data type	Description
	Returns	int	
	crs_id	int	
	std_id	int	