# irDevelopers.com [irDevelopers.com]

# Examination System Data Dictionary

2/24/2022

# Table of contents

xamin	ation System	7
1. Tab	oles	7
1.1.	Table: dbo.Choices	7
1.2.	Table: dbo.Course	7
1.3.	Table: dbo.Department	8
1.4.	Table: dbo.Ex_Qt	9
1.5.	Table: dbo.Exam	10
1.6.	Table: dbo.lns_Crs	11
1.7.	Table: dbo.Instructor	12
1.8.	Table: dbo.Question	13
1.9.	Table: dbo.St_Ex_Ch	14
1.10.	. Table: dbo.Std_Crs	15
1.11.	Table: dbo.Student	16
1.12.	. Table: dbo.Topic	16
1.13.	Table: dbo.Userr	17
2. Pro	ocedures	19
2.1.	Procedure: dbo.AddRelationInstructorToCourse	19
2.2.	Procedure: dbo.AddRelationStudentToCourse	19
2.3.	Procedure: dbo.CheckCredential	20
2.4.	Procedure: dbo.CorrectExam	20
2.5.	Procedure: dbo.CourseHasExams	21
2.6.	Procedure: dbo.delete_Department	21
2.7.	Procedure: dbo.delete_Std_Crs	22
2.8.	Procedure: dbo.deleteChoices	22
2.9.	Procedure: dbo.deleteCourse	23
2.10	). Procedure: dbo.deleteExam	23
2.11.	Procedure: dbo.deleteQuestion	24
2.12	Procedure: dbo.deleteStdSolveRelation	24
2.13	Procedure: dbo.deleteTopic	24
2.14	Procedure: dbo.deleteUserByEmail	25
2.15	Procedure: dbo.deleteUserByuserName	25
2.16	. Procedure: dbo.Exam_Answer	25
2.17	. Procedure: dbo.GenerateExam	27
2.18	Procedure: dbo.GetAllInstructorsForCourse	28
2.19	Procedure: dbo.GetAllStudentsForCourse	28
2.20	). Procedure: dbo.GetAviExams	29
2.21	Procedure: dbo.GetCoursesOfInstructor	29
2.22	2. Procedure: dbo.GETCourseTopics	30
2.23	3. Procedure: dbo.getExamQuestions	30
2.24	4. Procedure: dbo.getGradesWithID	31
2.25	5. Procedure: dbo.GETInstructorCourse	31
2.26	5. Procedure: dbo.GetInstructorDetails	32
2.27	7. Procedure: dbo.GetQuestionsWithChoices	32
2.28	3. Procedure: dbo.GetStudentDEtails	33
2.29	9. Procedure: dbo.GetStudentDtails	33

	Procedure: dbo.GETStudentGrades	
2.31.	Procedure: dbo.GETStudentInDep	34
2.32.	Procedure: dbo.GetTFQuestionForCourse	34
2.33.	Procedure: dbo.GetUserID	
2.34.	Procedure: dbo.GetUserIDAndType	
2.35.	Procedure: dbo.insert_Department	35
2.36.	Procedure: dbo.insert_Std_Crs	
2.37.	Procedure: dbo.InsertChoices	
2.38.	Procedure: dbo.insertCourse	
2.39.	Procedure: dbo.insertCourseWithInstrucot	
2.40.	Procedure: dbo.insertCourseWithInstructor	
2.41.	Procedure: dbo.insertExam	38
2.42.	Procedure: dbo.insertIns_Crs	38
2.43.	Procedure: dbo.insertInstructor	
2.44.	Procedure: dbo.insertQuestion	40
2.45.	Procedure: dbo.insertStdSolveRelation	40
2.46.	Procedure: dbo.insertStudent	40
2.47.	Procedure: dbo.insertTopic	41
2.48.	Procedure: dbo.insertUser	42
2.49.	Procedure: dbo.select_Department	42
2.50.	Procedure: dbo.select_Std_Crs	43
2.51.	Procedure: dbo.SetDegree	43
2.52.	Procedure: dbo.spGenerateDBDictionary	44
2.53.	Procedure: dbo.StudentQuestionsAnswers	44
2.54.	Procedure: dbo.up_generate_data_dictionary	45
2.55.	Procedure: dbo.update_Department	
2.56.	Procedure: dbo.update_Std_Crs	47
2.57.	Procedure: dbo.updateChoices	47
2.58.	Procedure: dbo.updateCourse	48
2.59.	Procedure: dbo.updateExma	48
2.60.	Procedure: dbo.updateinstructorSalary	49
2.61.	Procedure: dbo.updateQuestion	49
2.62.	Procedure: dbo.updateStudentGradeDate	50
2.63.	Procedure: dbo.updateTopic	50
2.64.	Procedure: dbo.UpdateUser	51
2.65.	Procedure: dbo.updateUserDepartment	51

# Legend

- **?** Primary key
- Primary key disabled
- 1 User-defined primary key
- ¶ Unique key
- ¶ Unique key disabled
- 1 User-defined unique key
- Active trigger
- Disabled trigger
- → Many to one relation
- $\succ_{\mathbf{i}}$  User-defined many to one relation
- ✓ User-defined one to many relation
- 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

# 1. Tables

# 1.1. Table: dbo.Choices

## Columns

	Name	Data type	Description / Attributes
1	q_ID	int	References: dbo.Question
1	choice1	nvarchar(20)	
	choice2	nvarchar(20)	
	choice3	nvarchar(20)	Nullable
	choice4	nvarchar(20)	Nullable

# Links to

	Table	Join	Title / Name / Description
>	dbo.Question	dbo.Choices.q_ID = dbo.Question.questionID	FK_Choices_q_ID_5441852A

# Unique keys

	Columns	Name / Description
Ŷ	choice1, q_ID	PK_Choices6C6330E59D1B84A2

# Uses

	Name		
<b></b>	dbo.Choices		
	→ dbo.Question		

# Used By

	Name
🌣 dbo.deleteChoices	
🌣 dbo.getExamQuestions	
abo.GetQuestionsWithChoices	
🌣 dbo.InsertChoices	
🌣 dbo.updateChoices	

# 1.2. Table: dbo.Course

# Columns

	Name	Data type	Description / Attributes
1	cID	int	Identity / Auto increment
1	cName	nvarchar(20)	
	duration	int	Nullable

# Linked from

	Table	Join	Title / Name / Description
$\rightarrow$	dbo.Exam	<b>dbo.Course.</b> cID = dbo.Exam.cID	FK_Exam_Course
$\rightarrow$	dbo.lns_Crs	dbo.Course.cID = dbo.Ins_Crs.crs_ID	FK_Ins_Crs_crs_ID_4BAC3F29
$\rightarrow$	dbo.Question	<b>dbo.Course.</b> cID = dbo.Question.cID	FK_Question_Course
$\rightarrow$	dbo.Std_Crs	dbo.Course.cID = dbo.Std_Crs.crs_ID	FK_Std_Crs_crs_ID_4F7CD00D
$\rightarrow$	dbo.Topic	<b>dbo.Course.</b> cID = dbo.Topic.crs_ID	FK_Topic_crs_ID_440B1D61

# Unique keys

	Columns Name / Description	
<b>?</b> cID PK_Course_D830D457A3417035		PK_Course_D830D457A3417035
Ŷ	CName UQ_Course_829A49F6664D5A38	

# Used By

	Name
Ⅲ dbo.Cc	ourse
🌣 dbo	o.deleteCourse
🌣 dbo	o.GetAviExams
🌣 dbo	o.GetCoursesOfInstructor
🌣 dbo	o.GETCourseTopics
🌣 dbo	o.getGradesWithID
🏖 dbo	o.GETInstructorCourse
🏖 dbo	o.GETStudentGrades
🏖 dbo	n.insertCourse
🏖 dbo	o.updateCourse
→ dbo	D.Exam
→ dbo	o.lns_Crs
→ dbo	o.Question
→ dbo	o.Std_Crs
→ dbo	р.Торіс

# 1.3. Table: dbo.Department

# Columns

		Name	Data type	Description / Attributes
	1	depID	int	Identity / Auto increment
	1	depName	nvarchar(20)	
		depDescription	nvarchar(50)	Nullable
181		ins_ID	int	Nullable References: dbo.Instructor

# Links to

	Table	Join	Title / Name / Description
<b>-</b>	dbo.Instructor	dbo.Department.ins_ID = dbo.Instructor.insID	c4

# Linked from

	Table	Join	Title / Name / Description
-		<b>dbo.Department</b> .depID = dbo.Userr.depID	FK_Userr_Department

# Unique keys

Columns Name / Description		Name / Description
P	PK_Departme_00D7A29357809BC9	
P	depName	UQ_Departme_49814543354FF10E

# Uses

	Name
<b>==</b>	dbo.Department
	→ dbo.Instructor

# Used By

	Name		
	■ dbo.Department		
1	b dbo.delete_Department		
1	dbo.insert_Department		
1	dbo.select_Department		
1	dbo.update_Department		
	→ dbo.Userr		

# 1.4. Table: dbo.Ex\_Qt

# Columns

	Name	Data type	Description / Attributes
1	Exam_ID	int	References: dbo.Exam
1	questionID	int	References: dbo.Question

# Links to

	Table	Join	Title / Name / Description
<b>-</b>	dbo.Exam	dbo.Ex_Qt.Exam_ID = dbo.Exam.Exam_ID	FK_Ex_Qt_Exam
$\rightarrow$	dbo.Question	dbo.Ex_Qt.questionID = dbo.Question.questionID	FK_Ex_Qt_Question

# Unique keys

	Columns	Name / Description
9	Exam_ID, questionID	PK_Ex_Qt

#### Uses

0363		
	Name	
Ⅲ dbo.Ex_Qt		
→ dbo.Exam		
→ dbo.Question		

# Used By

	Name
<b></b>	dbo.Ex_Qt
	dbo.Exam_Answer
	dbo.GenerateExam
	dbo.getExamQuestions
	dbo.StudentQuestionsAnswers

# 1.5. Table: dbo.Exam

# Columns

	Name	Data type	Description / Attributes
1	Exam_ID	int	Identity / Auto increment
	Exam_Date	date	Nullable
	cID	int	Nullable References: dbo.Course

# Links to

	Table	Join	Title / Name / Description
$\rightarrow$	dbo.Course	<b>dbo.Exam.</b> cID = dbo.Course.cID	FK_Exam_Course

# Linked from

	Table	Join	Title / Name / Description
$\rightarrow$	dbo.Ex_Qt	dbo.Exam.Exam_ID = dbo.Ex_Qt.Exam_ID	FK_Ex_Qt_Exam
$\rightarrow$	dbo.St_Ex_Ch	dbo.Exam.Exam_ID = dbo.St_Ex_Ch.Exam_ID	FK_St_Ex_Ch_Exam_I_59FA5E80

# Unique keys

	Columns	Name / Description
9	Exam_ID	PK_Exam_C782CA79D2302210

# Uses

Name
 dbo.Exam
→ dbo.Course

# Used By

	Name
■	dbo.Exam
	🏠 dbo.CourseHasExams
	🌣 dbo.deleteExam
	🏠 dbo.GenerateExam
	🏕 dbo.GetAviExams
	dbo.getExamQuestions
	🌣 dbo.insertExam
	🏠 dbo.SetDegree
	🌣 dbo.updateExma
	→ dbo.Ex_Qt
	$\rightarrow$ dbo.St_Ex_Ch

# 1.6. Table: dbo.lns\_Crs

# Columns

		Name	Data type	Description / Attributes
	1	ins_ID	int	References: dbo.lnstructor
[8]	1	crs_ID	int	References: dbo.Course

## Links to

	Table	Join	Title / Name / Description
<b>—</b>	dbo.Course	dbo.lns_Crs.crs_ID = dbo.Course.cID	FK_Ins_Crs_crs_ID_4BAC3F29
<b>-</b>	dbo.Instructor	dbo.lns_Crs.ins_ID = dbo.lnstructor.insID	FK_Ins_Crs_ins_ID_4AB81AF0

# Unique keys

	Columns	Name / Description
?	ins_ID, crs_ID	PK_Ins_Crs_527ACC3FE2DE0147

03	
	Name
	dbo.lns_Crs
	→ dbo.Course
	→ dbo.Instructor

# Used By

	Name
≡	dbo.lns_Crs
	bo.AddRelationInstructorToCourse
	dbo.GetAllInstructorsForCourse
	dbo.GetCoursesOfInstructor
	dbo.GETInstructorCourse
	dbo.insertIns_Crs

# 1.7. Table: dbo.Instructor

# Columns

	Name	Data type	Description / Attributes
1	insID	int	References: dbo.Userr
	salary	money	Nullable

# Links to

	Table	Join	Title / Name / Description
3	→ dbo.Userr	dbo.Instructor.insID = dbo.Userr.id	c3

# Linked from

	Table	Join	Title / Name / Description
$\rightarrow$	dbo.Department	dbo.Instructor.insID = dbo.Department.ins_ID	c4
$\rightarrow$	dbo.lns_Crs	dbo.Instructor.insID = dbo.Ins_Crs.ins_ID	FK_Ins_Crs_ins_ID_4AB81AF0

# Unique keys

	Columns	Name / Description
?	insID	PK_Instruct_117C668BAD440B38

## Uses

	Name
→ dbo.Userr	

	Name		
<b>=</b>	dbo.Instructor		
1	dbo.GetAllInstructorsForCourse		
1	dbo.GetCoursesOfInstructor		
1	dbo.GETInstructorCourse		
1	b dbo.GetInstructorDetails		
1	dbo.insertInstructor		

Name
b dbo.updateinstructorSalary
→ dbo.Department
→ dbo.lns_Crs

# 1.8. Table: dbo.Question

## Columns

		Name	Data type	Description / Attributes
	1	questionID	int	Identity / Auto increment
		questionText	nvarchar(150)	
		answer	nvarchar(1)	
		qТуре	nvarchar(3)	
[8]		cID	int	Nullable References: dbo.Course

# Links to

	Table	Join	Title / Name / Description
$\rightarrow$	dbo.Course	<b>dbo.Question.</b> cID = dbo.Course.cID	FK_Question_Course

# Linked from

	Table	Join	Title / Name / Description
$\rightarrow$	dbo.Choices	dbo.Question.questionID = dbo.Choices.q_ID	FK_Choices_q_ID_5441852A
$\rightarrow$	dbo.Ex_Qt	dbo.Question.questionID = dbo.Ex_Qt.questionID	FK_Ex_Qt_Question
$\rightarrow$	dbo.St_Ex_Ch	dbo.Question.questionID = dbo.St_Ex_Ch.questionID	FK_St_Ex_Ch_questi_5AEE82B9

# Unique keys

	Columns	Name / Description
9	questionID	PK_Question_6238D492847A7F74

# Uses

	Name
<b>==</b>	dbo.Question
3	→ dbo.Course

	Name
<b>==</b>	dbo.Question
1	dbo.CorrectExam
1	dbo.deleteQuestion
1	dbo.GenerateExam
4	dbo.getExamQuestions

	Name
abo.GetQuestionsWithChoices	
abo.GetTFQuestionForCourse	
dbo.InsertChoices	
dbo.insertQuestion	
abo.StudentQuestionsAnswers	
dbo.updateQuestion	
→ dbo.Choices	
→ dbo.Ex_Qt	
→ dbo.St_Ex_Ch	

# 1.9. Table: dbo.St\_Ex\_Ch

# Columns

	Name	Data type	Description / Attributes
1	Std_ID	int	References: dbo.Student
1	Exam_ID	int	References: dbo.Exam
1	questionID	int	References: dbo.Question
	St_Answer	nvarchar(1)	Nullable

# Links to

	Table	Join	Title / Name / Description
<b>-</b>	dbo.Exam	dbo.St_Ex_Ch.Exam_ID = dbo.Exam.Exam_ID	FK_St_Ex_Ch_Exam_I_59FA5E80
<b>&gt;</b>	dbo.Question	dbo.St_Ex_Ch.questionID = dbo.Question.questionID	FK_St_Ex_Ch_questi_5AEE82B9
<b>—</b>	dbo.Student	dbo.St_Ex_Ch.Std_ID = dbo.Student.stuID	FK_St_Ex_Ch_Std_ID_59063A47

# Unique keys

	Columns	Name / Description
1	Std_ID, Exam_ID, questionID	PK_St_Ex_ChE13150DD47DE2FC6

# Uses

	Name	
Ⅲ dbo.St_Ex_Ch		
→ dbo.Exam		
→ dbo.Question		
→ dbo.Student		

,	
	Name
dbo.St_Ex_Ch     dbo.St_Ex_Ch	
🏖 dbo.CorrectExam	

	Name
dbo.deleteStdSolveRelation	
dbo.Exam_Answer	
dbo.GetAviExams	
b dbo.insertStdSolveRelation	
abo.StudentQuestionsAnswers	

# 1.10. Table: dbo.Std\_Crs

# Columns

	Name	Data type	Description / Attributes
1	std_ID	int	References: dbo.Student
1	crs_ID	int	References: dbo.Course
	grade	float	Nullable

# Links to

	Table	Join	Title / Name / Description
>	dbo.Course	dbo.Std_Crs.crs_ID = dbo.Course.cID	FK_Std_Crs_crs_ID_4F7CD00D
<b>-</b>	dbo.Student	dbo.Std_Crs.std_ID = dbo.Student.stuID	FK_Std_Crs_std_ID4E88ABD4

# Unique keys

	Columns	Name / Description
P	std_ID, crs_ID	PK_Std_Crs_C5E735C5E9C85E8B

#### Uses

5565		
	Name	
Ⅲ dbo.Std_Crs		
→ dbo.Course		
→ dbo.Student		

	Name
<b></b>	dbo.Std_Crs
	dbo.AddRelationStudentToCourse
	dbo.delete_Std_Crs
	dbo.GetAllStudentsForCourse
	dbo.GetAviExams
	dbo.getGradesWithID
	dbo.GETInstructorCourse
	dbo.GETStudentGrades
	dbo.insert_Std_Crs
	dbo.select_Std_Crs

Name	
dbo.update_Std_Crs	

# 1.11. Table: dbo.Student

# Columns

	Name	Data type	Description / Attributes
1	stuID	int	References: dbo.Userr
	gradYear	date	Nullable

#### Links to

	Table	Join	Title / Name / Description
$\rightarrow$	dbo.Userr	dbo.Student.stuID = dbo.Userr.id	c2

## Linked from

	Table	Join	Title / Name / Description
<b>→</b>	dbo.St_Ex_Ch	dbo.Student.stuID = dbo.St_Ex_Ch.Std_ID	FK_St_Ex_Ch_Std_ID_59063A47
<b>→</b>	dbo.Std_Crs	dbo.Student.stuID = dbo.Std_Crs.std_ID	FK_Std_Crs_std_ID_4E88ABD4

# Unique keys

	Columns	Name / Description
P	stuID	PK_Student_AEC9BFAF96711633

# Uses

	Name
-	dbo.Student
:	→ dbo.Userr

# Used By

	Name
	dbo.Student
1	dbo.GetAllStudentsForCourse
1	dbo.GetStudentDEtails
1	dbo.GetStudentDtails
1	dbo.GETStudentGrades
1	dbo.insertStudent
1	dbo.updateStudentGradeDate
-	→ dbo.St_Ex_Ch
-	→ dbo.Std_Crs

# 1.12. Table: dbo.Topic

# Columns

	Name	Data type	Description / Attributes
1	topicID	int	Identity / Auto increment
	topicName	nvarchar(20)	
	crs_ID	int	Nullable References: dbo.Course

# Links to

	Table	Join	Title / Name / Description
<b>&gt;</b>	dbo.Course	dbo.Topic.crs_ID = dbo.Course.cID	FK_Topic_crs_ID_440B1D61

# Unique keys

	Columns	Name / Description
Ŷ	topicID	PK_Topic_72C15B21DEDA84B0

# Uses

	Name
dbo.Topic	
→ dbo.Course	

# Used By

	Name
Ⅲ dbo.Top	oic .
🌣 dbo.	deleteTopic
🌣 dbo.	GETCourseTopics
🌣 dbo.i	insertTopic
🌣 dbo.	updateTopic

# 1.13. Table: dbo.Userr

# Columns

	Name	Data type	Description / Attributes
1	id	int	Identity / Auto increment
	firstName	nvarchar(20)	Nullable
	lastName	nvarchar(20)	Nullable
1	email	nvarchar(15)	
1	userName	nvarchar(15)	
	passKey	nvarchar(15)	
	sex	nvarchar(1)	Nullable
	userType	nvarchar(1)	Nullable
	depID	int	Nullable References: dbo.Department

# Links to

		Table	Join	Title / Name / Description
3	→	dbo.Department	dbo.Userr.depID = dbo.Department.depID	FK_Userr_Department

# Linked from

	Table	Join	Title / Name / Description
$\rightarrow$	dbo.Instructor	<b>dbo.Userr.</b> id = dbo.Instructor.insID	3
$\rightarrow$	dbo.Student	dbo.Userr.id = dbo.Student.stuID	c2

# Unique keys

	Columns	Name / Description	
Ŷ	id	PK_Userr_3213E83FEB6608C1	
Ŷ	userName	UQ_Userr66DCF95CFCD60876	
Ŷ	email	UQ_Userr_AB6E616405BFD6A9	

# Uses

	Name
-	dbo.Userr
3	→ dbo.Department

used by	Name
dbo.CheckCredential	
dbo.deleteUserByEmail	
b dbo.deleteUserByuserName	
abo.GetAllInstructorsForCourse	
abo.GetAllStudentsForCourse	
dbo.GetInstructorDetails	
b dbo.GetStudentDEtails	
b dbo.GetStudentDtails	
🍇 dbo.GETStudentInDep	
dbo.GetUserID	
dbo.GetUserIDAndType	
dbo.insertInstructor	
dbo.insertStudent	
dbo.insertUser	
abo.updateinstructorSalary	
abo.updateStudentGradeDate	
dbo.UpdateUser	
dbo.updateUserDepartment	

Name
→ dbo.Instructor
→ dbo.Student

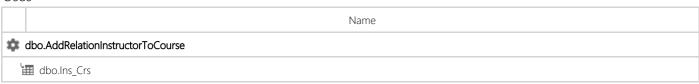
# 2. Procedures

#### 2.1. Procedure: dbo.AddRelationInstructorToCourse

## Input/Output

	Name	Data type	Description
<b>→</b> @	CID	int	
<b>→</b> @	InsID	int	
→@	STATE	int	

#### Uses



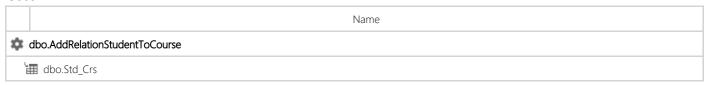
#### Script

```
create    proc AddRelationInstructorToCourse @CID int ,@InsID int,@STATE Int
    as
    begin
    if(@STATE =1)
    begin
        insert into Ins_Crs(crs_ID,ins_ID)values (@CID,@InsID)
    end
    else
        delete from Ins_Crs where crs_ID = @CID and ins_ID =@InsID
    end
```

#### 2.2. Procedure: dbo.AddRelationStudentToCourse

#### Input/Output

	Name	Data type	Description
→@	CID	int	
<b>→</b> @	StID	int	
<b>→</b> @	STATE	int	



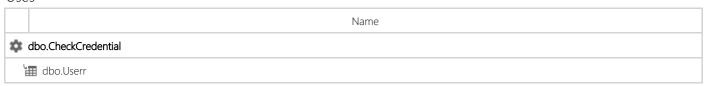
```
create proc AddRelationStudentToCourse @CID int ,@StID int,@STATE Int
    as
    begin
    if(@STATE =1)
    begin
        insert into Std_Crs(crs_ID,std_ID)values (@CID,@StID)
    end
    else
        delete from Std_Crs where crs_ID = @CID and std_ID =@StID
    end
end
```

#### 2.3. Procedure: dbo.CheckCredential

#### Input/Output

	Name	Data type	Description
<b>→</b> @	userName	nvarchar(15)	
<b>→</b> @	passKey	nvarchar(15)	
÷@>	isAuth	bit	

#### Uses



## Script

```
CREATE PROC CheckCredential @userName nvarchar(15), @passKey nvarchar(15), @isAuth BIT OUT AS
BEGIN

IF EXISTS(SELECT ID FROM Userr

WHERE userName = @userName AND passKey = @passKey)

SET @isAuth = 1

ELSE

SET @isAuth = 0

END
```

#### 2.4. Procedure: dbo.CorrectExam

#### Input/Output

	Name	Data type	Description
<b>→</b> @	ExamID	int	
<b>→</b> @	StuID	int	
÷@•	GRADE	nvarchar(1)	

	Nan	ne
:0:	dbo.CorrectExam	
Ĺ	dbo.Question	
Ĺ	dbo.St_Ex_Ch	

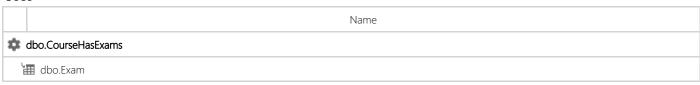
```
CREATE
            PROCEDURE CorrectExam @ExamID INT, @StuID INT, @GRADE nvarchar(1) OUT
           BEGIN
                DECLARE
                     @St_Answer NVARCHAR(1),
                     @actual answer NVARCHAR(1),
                SET @GRADE = 0;
                DECLARE cursor_answers CURSOR
                FOR SELECT St_Answer, [dbo].[Question].answer FROM [dbo].[St_Ex_Ch]
INNER JOIN [dbo].[Question] ON [dbo].[Question].questionID = [dbo].[St_Ex_Ch].questionID
AND [dbo].[St_Ex_Ch].Exam_ID = @ExamID AND [dbo].[St_Ex_Ch].Std_ID = @StuID;
                OPEN cursor_answers;
                FETCH NEXT FROM cursor answers INTO
                     @St_Answer,
                     @actual_answer;
                WHILE @@FETCH STATUS = 0
                           IF (@St_Answer = @actual_answer)
    SET @GRADE = @GRADE + 1;
                           FETCH NEXT FROM cursor_answers INTO
                                 @St Answer,
                                @actual_answer;
                     END;
                CLOSE cursor_answers;
DEALLOCATE cursor_answers;
                SELECT @CID = [dbo].[Question].cID FROM [dbo].[St_Ex_Ch]
                INNER JOIN [dbo].[Question] ON [dbo].[Question].questionID = [dbo].[St_Ex_Ch].questionID AND [dbo].[St_Ex_Ch].Exam_ID = @ExamID AND [dbo].[St_Ex_Ch].Std_ID = @StuID;
                EXECUTE update_Std_Crs @StuID, @CID, @GRADE
           END
```

#### 2.5. Procedure: dbo.CourseHasExams

#### Input/Output

	Name	Data type	Description
<b>→</b> @	CID	int	

#### Uses



## Script

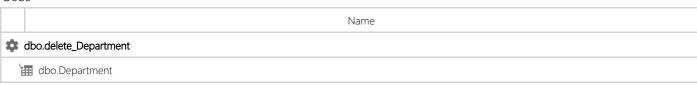
```
create    proc CourseHasExams @CID int
as
select count (*) from Exam e where e.cID =@CID
```

#### 2.6. Procedure: dbo.delete\_Department

#### Input/Output

	Name	Data type	Description
<b>→</b> @	DeptID	int	

#### Uses



#### Script

```
CREATE PROC delete_Department @DeptID int
AS

BEGIN TRY

DELETE FROM Department
WHERE depID = @DeptID

END TRY
BEGIN CATCH

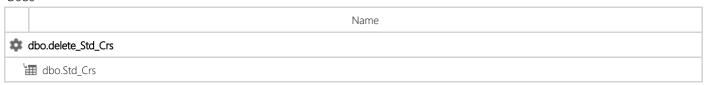
SELECT 'Falied To delete!'
END CATCH
```

# 2.7. Procedure: dbo.delete\_Std\_Crs

## Input/Output

	Name	Data type	Description
<b>→</b> @	StudentID	int	
<b>→</b> @	CourselD	int	

#### Uses



#### Script

```
CREATE PROC delete_Std_Crs @StudentID int, @CourseID int
AS

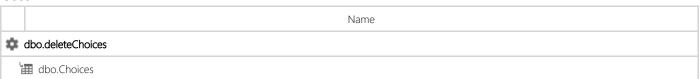
BEGIN TRY

DELETE FROM Std_Crs
WHERE std_ID = @StudentID AND crs_ID = @CourseID
END TRY
BEGIN CATCH
SELECT 'Falied To delete!'
END CATCH
```

#### 2.8. Procedure: dbo.deleteChoices

## Input/Output

	Name	Data type	Description
<b>→</b> @	questionID	int	



```
CREATE PROC deleteChoices @questionID INT

AS

BEGIN TRY

EXECUTE deleteQuestion @questionID

DELETE FROM Choices
WHERE q_ID = @questionID

END TRY
BEGIN CATCH

SELECT 'Falied To delete!'

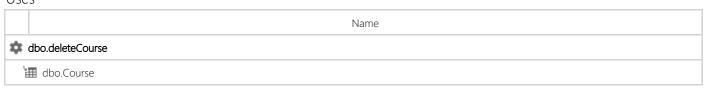
END CATCH
```

#### 2.9. Procedure: dbo.deleteCourse

#### Input/Output

	Name	Data type	Description
→@	courseID	int	

#### Uses



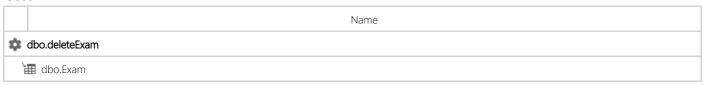
## Script

#### 2.10. Procedure: dbo.deleteExam

#### Input/Output

	Name	Data type	Description
<b>→</b> @	Exam_ID	int	

#### Uses



#### Script

```
-- DELETE
CREATE PROC deleteExam @Exam_ID INT
AS

BEGIN TRY

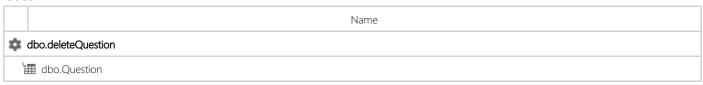
DELETE FROM Exam
WHERE Exam_ID = @Exam_ID
END TRY
BEGIN CATCH
SELECT 'Falied To delete!'
END CATCH
```

#### 2.11. Procedure: dbo.deleteQuestion

#### Input/Output

	Name	Data type	Description
→@	questionID	int	

#### Uses



#### Script

```
CREATE PROC deleteQuestion @questionID INT
AS

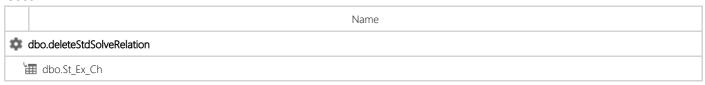
DELETE FROM Question
WHERE questionID = @questionID
```

# 2.12. Procedure: dbo.deleteStdSolveRelation

#### Input/Output

	Name	Data type	Description
→@	Std_ID	int	
→@	Exam_ID	int	
→@	questionID	int	

#### Uses



### Script

```
CREATE PROC deleteStdSolveRelation @Std_ID INT, @Exam_ID INT, @questionID INT

AS

BEGIN TRY

DELETE FROM St_Ex_Ch

WHERE @Std_ID = Std_ID AND @Exam_ID = Exam_ID AND @questionID = questionID

END TRY

BEGIN CATCH

SELECT 'Falied To delete!'

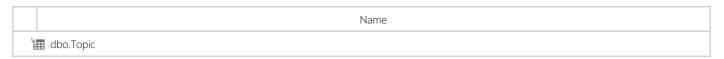
END CATCH
```

# 2.13. Procedure: dbo.deleteTopic

#### Input/Output

	Name	Data type	Description
<b>→</b> @	topic_ID	int	

	Name	
*	dbo.deleteTopic	

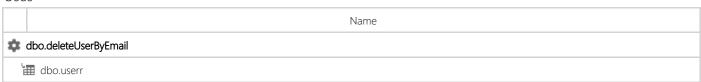


# 2.14. Procedure: dbo.deleteUserByEmail

#### Input/Output

	Name	Data type	Description
<b>→</b> @	email	nvarchar(15)	

#### Uses



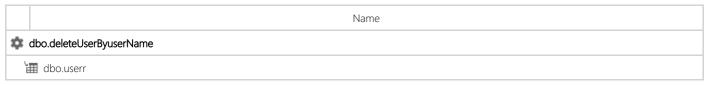
#### Script

# 2.15. Procedure: dbo.deleteUserByuserName

#### Input/Output

	Name	Data type	Description
<b>→</b> @	userName	nvarchar(15)	

#### Uses



#### Script

## 2.16. Procedure: dbo.Exam\_Answer

# Input/Output

	Name	Data type	Description
<b>→</b> @	ExamID	int	
<b>→</b> @	StuID	int	
<b>→</b> @	ans1	nvarchar(1)	
<b>→</b> @	ans2	nvarchar(1)	
<b>→</b> @	ans3	nvarchar(1)	
<b>→</b> @	ans4	nvarchar(1)	
<b>→</b> @	ans5	nvarchar(1)	
<b>→</b> @	ans6	nvarchar(1)	
<b>→</b> @	ans7	nvarchar(1)	
<b>→</b> @	ans8	nvarchar(1)	
<b>→</b> @	ans9	nvarchar(1)	
→@	ans10	nvarchar(1)	

	Name
*	dbo.Exam_Answer
١	dbo.ex_qt
١	dbo.St_Ex_Ch

```
PROCEDURE Exam Answer @ExamID int, @StuID int, @ans1 nvarchar(1)
, @ans2 nvarchar(1) , @ans3 nvarchar(1) , @ans4 nvarchar(1) , @ans5 nvarchar(1) , @ans6 nvarchar(1) , @ans7 nvarchar(1) , @ans8 nvarchar(1) , @ans9 nvarchar(1) , @ans10 nvarchar(1)
        begin
                                    declare @counter int = 1
                                    declare c1 cursor
                                    for select questionID
                                    from ex_qt
                                    where Exam_ID = @ExamID
                        for read only
                        declare @QuestionID int
                        declare @count int
                        set @count =0
                        open cl
                       fetch c1 into @QuestionID
while @@FETCH_STATUS=0
begin
                                    if @count =0
                                                insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans1)
                                    else if @count=1
                                                insert into St Ex Ch values (@StuID, @ExamID, @QuestionID, @ans2)
                                    else if @count=2
                                                insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans3)
                                    else if @count=3
                                                insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans4)
                                    else if @count=4
                                                insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans5)
                                    else if @count=5
                                                insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans6)
                                    else if @count=6
                                                insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans7)
                                    else if @count=7
                                    insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans8)
                                    else if @count=8
                                    insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans9)
                                               insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans10)
                                    set @count =@count+1
                            fetch cl into @QuestionID
                                   end
                                    close c1
                                    deallocate c1
        end
```

#### 2.17. Procedure: dbo.GenerateExam

#### Input/Output

	Name	Data type	Description
→@	CourseID	int	
→@	NumTF	int	
→@	NumMC	int	

	Name
<b>☆</b> dbo.GenerateExam	
dbo.Ex_Qt	
dbo.Exam	
dbo.Question	

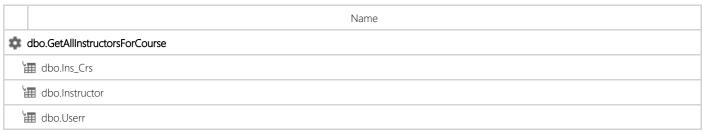
```
CREATE
           PROCEDURE [dbo].[GenerateExam] @CourseID int, @NumTF int, @NumMC int
     IF @NumTF + @NumMC > 10
         BEGIN
              select 'error'
         END
         begin
                                         declare @Date date = DATEADD(month,2, getdate());
             INSERT INTO Exam (Exam_Date ,cID)
VALUES (@Date,@CourseID)
                                          declare @Exam ID int = IDENT CURRENT ('Exam')
                                          select @Exam_ID
                                          declare @mcqIDs table(id int) ;
                                          insert into @mcqIDs select top(@NumMC)q.questionID
                                          from Question q where q.qType = 'mcq' order by NEWID();
declare @tfIDs table(id int);
insert into @tfIDs select top(@NumMC)q.questionID
                                          from Question q where q.qType = 'T/F' order by NEWID(); insert into Ex_Qt select @Exam_ID ,q.id from @tfIDs q;
                                          insert into Ex_Qt select @Exam_ID ,q.id from @mcqIDs q;
          end
```

#### 2.18. Procedure: dbo.GetAllInstructorsForCourse

#### Input/Output

	Name	Data type	Description
<b>→</b> @	CrsID	int	

#### Uses



#### Script

```
CREATE PROCEDURE GetAllInstructorsForCourse @CrsID int

AS

BEGIN

SELECT id, firstName + ' ' + lastName as fullName, email, userName,

CASE WHEN crs_ID IS NULL THEN 0 ELSE 1 END as Assign

FROM Userr INNER JOIN Instructor

ON Userr.id = Instructor.insID

LEFT JOIN Ins_Crs

ON Ins_Crs.crs_ID = @CrsID

AND Instructor.insID = Ins_Crs.ins_ID

END
```

#### 2.19. Procedure: dbo.GetAllStudentsForCourse

#### Input/Output

	Name	Data type	Description
<b>→</b> @	CrsID	int	



```
Name

dbo.Student

dbo.Userr
```

```
CREATE PROCEDURE GetAllStudentsForCourse @CrsID int

AS

BEGIN

SELECT id, firstName + ' ' + lastName as fullName, email, userName,

CASE WHEN crs_ID IS NULL THEN 0 ELSE 1 END as enrolled

FROM Userr INNER JOIN Student

ON Userr.id = Student.stuID

LEFT JOIN Std_Crs
ON Std_Crs.crs_ID = @CrsID
AND Std_Crs.std_ID = Student.stuID

END
```

#### 2.20. Procedure: dbo.GetAviExams

#### Input/Output

	Name	Data type	Description
<b>→</b> @	STD_ID	int	

#### Uses

	Name
dbo.GetAviExams	
dbo.COURSE	
dbo.EXAM	
dbo.St_Ex_Ch	
dbo.Std_Crs	

#### Script

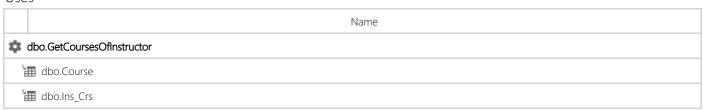
```
CREATE PROC GetAviExams @STD_ID INT
AS
BEGIN

SELECT Exam_ID, cName FROM Std_Crs
INNER JOIN COURSE CRS
ON Std_Crs.crs_ID = CRS.cID AND std_ID = @STD_ID
INNER JOIN EXAM
ON EXAM.cID = CRS.cID
WHERE NOT EXISTS(SELECT * FROM St_Ex_Ch S WHERE S.Std_ID = @STD_ID AND S.Exam_ID = Exam.Exam_ID)
END
```

#### 2.21. Procedure: dbo.GetCoursesOfInstructor

#### Input/Output

	Name	Data type	Description
<b>→</b> @	ins_id	int	



	Name	
ı	dbo.Instructor	

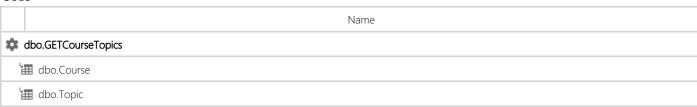
```
create    proc GetCoursesOfInstructor @ins_id int
as
select c.* from Instructor i inner join Ins_Crs ic on i.insID =ic.ins_ID and i.insID =@ins_id
inner join Course c on ic.crs_ID= c.cID
```

# 2.22. Procedure: dbo.GETCourseTopics

# Input/Output

	Name	Data type	Description
<b>→</b> @	CID	int	

#### Uses



#### Script

```
CREATE PROCEDURE GETCourseTopics @CID INT

As

BEGIN

select t.* from Course c inner join Topic t on t.crs_ID = c.cID and c.cID = @CID

End
```

# 2.23. Procedure: dbo.getExamQuestions

#### Input/Output

	Name	Data type	Description
<b>→</b> @	Exam_id	int	

0363	
	Name
dbo.getExamQuestions	
dbo.Choices	
dbo.Ex_Qt	
dbo.Exam	
dbo.Question	

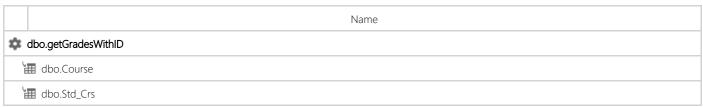
```
CREATE PROC GetExamQuestions @Exam id INT
BEGIN
            DECLARE @table TABLE (questionText nvarchar(150),questionID INT, qType nvarchar(3), choice1 nvarchar(20), choice2
nvarchar(20), choice3 nvarchar(20), choice4 nvarchar(20));
            INSERT INTO @table (questionText, questionID, qType, choice1, choice2, choice3, choice4)
            SELECT questionText, QT.questionID, qType, choice1, choice2, choice3, choice4
            FROM Exam ex
            INNER JOIN Ex_Qt
ON ex.Exam_ID = Ex_Qt.Exam_ID AND @Exam_id = EX.Exam_ID
            INNER JOIN Question qt
            ON qt.questionID = Ex_Qt.questionID AND qt.qType ='MCQ'
            INNER JOIN Choices ch
            ON qt.questionID = ch.q_ID
            INSERT INTO @table (questionText, questionID, qType, choice1, choice2, choice3, choice4)
SELECT questionText, QT.questionID, qType, 'True', 'False', '', ''
            FROM Exam ex
            INNER JOIN Ex_Qt
ON ex.Exam_ID = Ex_Qt.Exam_ID AND @Exam_id = EX.Exam_ID
            INNER JOIN Question qt
            ON qt.questionID = Ex_Qt.questionID AND qt.qType ='T/F'
            SELECT * FROM @table
END
```

#### 2.24. Procedure: dbo.getGradesWithID

#### Input/Output

	Name	Data type	Description
→@	std_id	int	

#### Uses

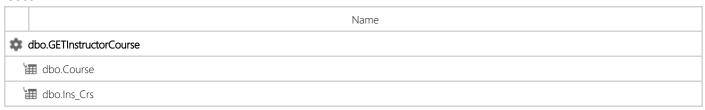


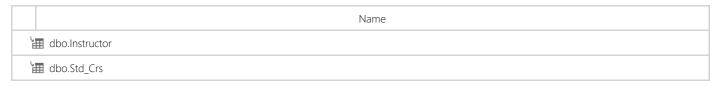
#### Script

#### 2.25. Procedure: dbo.GETInstructorCourse

#### Input/Output

	Name	Data type	Description
<b>→</b> @	insID	int	



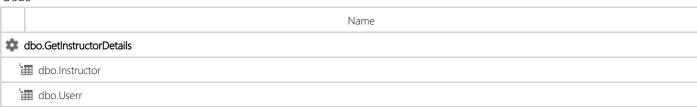


#### 2.26. Procedure: dbo.GetInstructorDetails

#### Input/Output

	Name	Data type	Description
<b>→</b> @	ins_id	int	

#### Uses



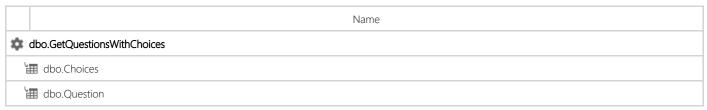
#### Script

## 2.27. Procedure: dbo.GetQuestionsWithChoices

#### Input/Output

	Name	Data type	Description
<b>→</b> @	Cld	int	

#### Uses



#### Script

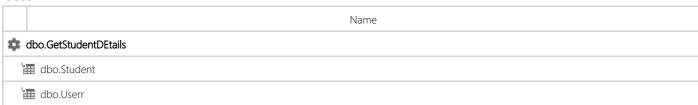
```
create    proc GetQuestionsWithChoices @CId int
as
select Q.* , ch.* from Question Q inner join Choices ch on Q.questionID = Ch.q_ID and Q.cID = @CId;
```

#### 2.28. Procedure: dbo.GetStudentDEtails

#### Input/Output

	Name	Data type	Description
<b>→</b> @	st_id	int	

#### Uses



#### Script

```
CREATE PROC GetStudentDetails @st_id INT
AS

BEGIN

SELECT * FROM Userr U

INNER JOIN Student S

ON U.ID = @st_id AND U.id = S.stuID
```

#### 2.29. Procedure: dbo.GetStudentDtails

#### Input/Output

	Name	Data type	Description
<b>→</b> @	st_id	int	

#### Uses

```
Name

dbo.GetStudentDtails

dbo.Student

dbo.Userr
```

#### Script

```
Create PROC GetStudentDtails @st_id INT
AS

BEGIN

SELECT * FROM Userr U

INNER JOIN Student S

ON U.ID = @st_id AND U.id = S.stuID

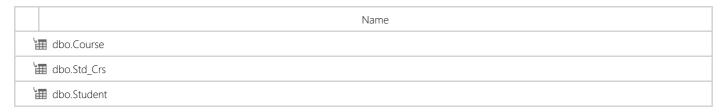
END
```

## 2.30. Procedure: dbo.GETStudentGrades

#### Input/Output

	Name	Data type	Description
<b>→</b> @	stulD	int	

	Name	
dbo.GETStudentGrades		



```
CREATE PROCEDURE [dbo].[GETStudentGrades] @stuID INT

As

BEGIN

select c.cName ,CONCAT(sc.grade*10,'%') AS Grade from Student s
inner join Std_Crs sc on sc.std_ID = s.stuID and s.stuID =@stuID
inner join Course c on sc.crs_ID = c.cID

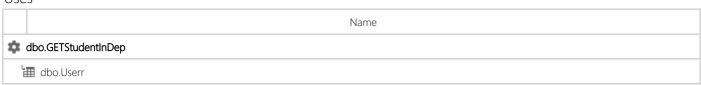
End
```

# 2.31. Procedure: dbo.GETStudentInDep

#### Input/Output

	Name	Data type	Description
<b>→</b> @	depID	int	

#### Uses



#### Script

```
CREATE PROCEDURE [dbo].[GETStudentInDep] @depID INT
As

BEGIN

select u.firstName,u.lastName,u.email,u.sex from Userr u where u.depID=@depID and u.userType

= 's';

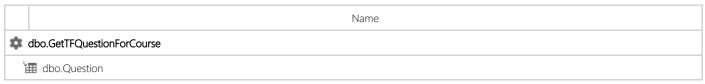
End
```

#### 2.32. Procedure: dbo.GetTFQuestionForCourse

#### Input/Output

	Name	Data type	Description
<b>→</b> @	CID	int	

#### Uses



#### Script

```
create    proc GetTFQuestionForCourse @CID int
as
select q.* from Question q where q.qType = 'T/f' and q.cID =@CID
```

#### 2.33. Procedure: dbo.GetUserID

# Input/Output

	Name	Data type	Description
<b>→</b> @	userName	nvarchar(15)	
<b>-</b> @•	ID	int	

#### Uses

	Name
<b>☆</b> dbo.GetUserID	
dbo.Userr	

# Script

```
CREATE PROC GetUserID @userName nvarchar(15), @ID INT OUT AS
BEGIN

SELECT @ID = ID FROM Userr

WHERE userName = @userName
END
```

# 2.34. Procedure: dbo.GetUserIDAndType

#### Input/Output

	Name	Data type	Description
<b>→</b> @	userName	nvarchar(15)	
÷@>	ID	int	
<b>-∕@</b> >	userType	nvarchar(1)	

#### Uses

	Name	
<b>☆</b> dbo.GetUserIDAndType		
Ę	dbo.Userr	

# Script

```
CREATE PROC GetUserIDAndType @userName nvarchar(15), @ID INT OUT, @userType nvarchar(1) OUT AS
BEGIN

SELECT @ID = ID, @userType = userType FROM Userr

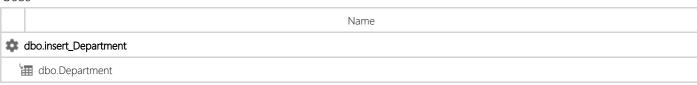
WHERE userName = @userName
END
```

# 2.35. Procedure: dbo.insert\_Department

## Input/Output

	Name	Data type	Description
<b>→</b> @	DeptName	nvarchar(20)	
<b>→</b> @	DeptDesc	nvarchar(50)	
<b>→</b> @	InstructorID	int	

#### Uses



#### Script

```
CREATE PROC [dbo].[insert_Department] @DeptName nvarchar(20), @DeptDesc nvarchar(50), @InstructorID int

AS

BEGIN TRY

INSERT INTO Department(depName, depDescription, ins_ID)

VALUES (@DeptName, @DeptDesc, @InstructorID)

END TRY

BEGIN CATCH

SELECT 'Failed to Insert!'

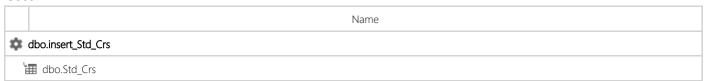
END CATCH
```

# 2.36. Procedure: dbo.insert\_Std\_Crs

#### Input/Output

	Name	Data type	Description
<b>→</b> @	StudentID	int	
<b>→</b> @	CourseID	int	
<b>→</b> @	grade	float	

#### Uses



#### Script

```
CREATE PROC insert_Std_Crs @StudentID int, @CourseID int, @grade float
AS

BEGIN TRY

INSERT INTO Std_Crs(std_ID, crs_ID, grade)

VALUES (@StudentID, @CourseID, @grade)

END TRY

BEGIN CATCH

SELECT 'Failed to Insert!'

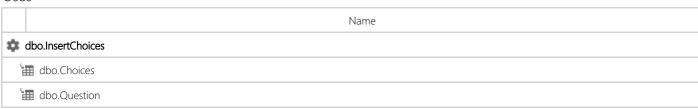
END CATCH
```

#### 2.37. Procedure: dbo.InsertChoices

#### Input/Output

	Name	Data type	Description
<b>→</b> @	q_ID	int	
<b>→</b> @	choice1	nvarchar(20)	
<b>→</b> @	choice2	nvarchar(20)	
<b>→</b> @	choice3	nvarchar(20)	
<b>→</b> @	choice4	nvarchar(20)	

#### Uses



#### Script

```
CREATE PROC InsertChoices @q_ID INT, @choice1 nvarchar(20), @choice2 nvarchar(20), @choice3 nvarchar(20), @choice4
nvarchar(20)

AS

BEGIN TRY

IF EXISTS (SELECT * FROM Question WHERE [questionID] = @q_ID)

INSERT INTO Choices VALUES (@q_ID, @choice1, @choice2, @choice3, @choice4)

ELSE

SELECT 'Failed To Insert!'

END TRY

BEGIN CATCH

SELECT 'Failed To Insert!'

END CATCH
```

### 2.38. Procedure: dbo.insertCourse

# Input/Output

	Name	Data type	Description
<b>→</b> @	cName	nvarchar(20)	
<b>→</b> @	duration	int	

#### Uses



## Script

## 2.39. Procedure: dbo.insertCourseWithInstrucot

# Input/Output

	Name	Data type	Description
<b>→</b> @	cName	nvarchar(20)	
→@	duration	int	
<b>→</b> @	Ins_ID	int	

## 2.40. Procedure: dbo.insertCourseWithInstructor

#### Input/Output

	Name	Data type	Description
→@	cName	nvarchar(20)	
→@	duration	int	
→@	Ins_ID	int	

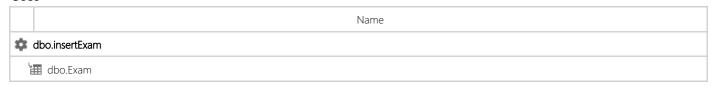
#### Script

#### 2.41. Procedure: dbo.insertExam

#### Input/Output

	Name	Data type	Description
<b>→</b> @	Exam_Date	date	
<b>→</b> @	CID	int	

## Uses



#### Script

```
CREATE PROC [dbo].[insertExam] @Exam_Date DATE ,@CID int
AS

BEGIN TRY

INSERT INTO Exam (Exam_Date ,cID)

VALUES (@Exam_Date,@CID)

END TRY
BEGIN CATCH

SELECT 'Failed to Insert!'

END CATCH
```

# 2.42. Procedure: dbo.insertIns\_Crs

# Input/Output

	Name	Data type	Description
<b>→</b> @	crs_ID	int	
<b>→</b> @	ins_ID	int	

#### Uses



## Script

#### 2.43. Procedure: dbo.insertInstructor

# Input/Output

	Name	Data type	Description
<b>→</b> @	fName	nvarchar(20)	
<b>→</b> @	IfName	nvarchar(20)	
<b>→</b> @	email	nvarchar(15)	
<b>→</b> @	userName	nvarchar(15)	
<b>→</b> @	password	nvarchar(15)	
<b>→</b> @	sex	nvarchar(1)	
<b>→</b> @	depld	int	
<b>→</b> @	sal	money	

#### Uses

00	5565	
	Name	
to dbo.insertInstructor		
	dbo.instructor	
	₩ dbo.userr	

## Script

## 2.44. Procedure: dbo.insertQuestion

# Input/Output

	Name	Data type	Description
<b>→</b> @	questionText	nvarchar(150)	
<b>→</b> @	answer	nvarchar(1)	
<b>→</b> @	qТуре	nvarchar(3)	
<b>→</b> @	CID	int	

#### Uses

	Name
*	dbo.insertQuestion
Ì	■ dbo.Question

#### Script

```
CREATE PROC [dbo].[insertQuestion] @questionText nvarchar(150), @answer nvarchar(1), @qType nvarchar(3) ,@CID int

AS

BEGIN TRY

INSERT INTO Question (questionText, answer, qType,cID)

VALUES (@questionText, @answer, @qType,@CID)

END TRY

BEGIN CATCH

SELECT 'Insertion Failed!'

END CATCH
```

## 2.45. Procedure: dbo.insertStdSolveRelation

## Input/Output

	Name	Data type	Description
<b>→</b> @	Std_ID	int	
<b>→</b> @	Exam_ID	int	
<b>→</b> @	questionID	int	
<b>→</b> @	St_Answer	nvarchar(1)	

#### Uses

```
Name

dbo.insertStdSolveRelation

dbo.St_Ex_Ch
```

## Script

```
CREATE PROC insertStdSolveRelation @Std_ID INT, @Exam_ID INT, @questionID INT, @St_Answer NVARCHAR(1)

AS

BEGIN TRY

INSERT INTO [dbo].[St_Ex_Ch]

VALUES (@Std_ID, @Exam_ID, @questionID, @St_Answer)

END TRY

BEGIN CATCH

SELECT 'Failed to Insert!'

END CATCH
```

# 2.46. Procedure: dbo.insertStudent

# Input/Output

	Name	Data type	Description
<b>→</b> @	fName	nvarchar(20)	
<b>→</b> @	IfName	nvarchar(20)	
<b>→</b> @	email	nvarchar(15)	
<b>→</b> @	userName	nvarchar(15)	
<b>→</b> @	password	nvarchar(15)	
<b>→</b> @	sex	nvarchar(1)	
<b>→</b> @	depld	int	
<b>→</b> @	gradeDate	date	

#### Uses

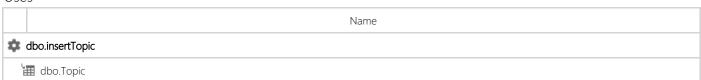
0303		
	Name	
state dbo.insertStudent		
dbo.student		
dbo.userr		

# Script

# 2.47. Procedure: dbo.insertTopic

# Input/Output

	Name	Data type	Description
<b>→</b> @	topicName	nvarchar(20)	
<b>→</b> @	courseID	int	



## 2.48. Procedure: dbo.insertUser

# Input/Output

	Name	Data type	Description
<b>→</b> @	fName	nvarchar(20)	
<b>→</b> @	IfName	nvarchar(20)	
<b>→</b> @	email	nvarchar(15)	
<b>→</b> @	userName	nvarchar(15)	
<b>→</b> @	password	nvarchar(15)	
<b>→</b> @	sex	nvarchar(1)	

## Uses

	Name
ıţ:	dbo.insertUser
ì	dbo.userr

# Script

```
CREATE proc insertUser @fName nvarchar(20),@lfName nvarchar(20),@email nvarchar(15),@userName nvarchar(15)
,@password nvarchar(15), @sex nvarchar(1)
as BEGIN TRY

INSERT INTO userr (firstName,lastName,email,userName,passKey,sex) values

(@fName,@lfName,@email,@userName,@password,@sex)

END TRY

BEGIN CATCH

SELECT 'Falied To Insert'

END CATCH
```

# 2.49. Procedure: dbo.select\_Department

# Input/Output

	Name	Data type	Description
<b>→</b> @	DeptID	int	

0.303	
	Name
dbo.select_Department	
dbo.Department	

```
CREATE PROC select_Department @DeptID int

AS

BEGIN TRY

SELECT *
FROM Department
WHERE depID = @DeptID

END TRY
BEGIN CATCH

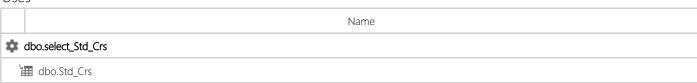
SELECT 'Falied To Get!'
END CATCH
```

# 2.50. Procedure: dbo.select\_Std\_Crs

# Input/Output

	Name	Data type	Description
<b>→</b> @	StudentID	int	
<b>→</b> @	CourseID	int	

#### Uses



# Script

```
CREATE PROC select_Std_Crs @StudentID int, @CourseID int

AS

BEGIN TRY

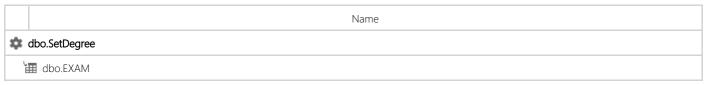
SELECT *
FROM Std_Crs
WHERE std_ID = @StudentID AND crs_ID = @CourseID

END TRY
BEGIN CATCH
SELECT 'Falied To Get!'
END CATCH
```

# 2.51. Procedure: dbo.SetDegree

# Input/Output

	Name	Data type	Description
→@	St_id	int	
→@	Exam_id	int	
→@	grade	float	



```
CREATE PROC SetDegree @St_id INT, @Exam_id INT, @grade FLOAT
AS
BEGIN

DECLARE @CID INT;

SELECT @CID = cID FROM EXAM
WHERE Exam_id = @Exam_id;

EXEC update_Std_Crs @St_id, @Exam_id, @grade
```

## 2.52. Procedure: dbo.spGenerateDBDictionary

#### Script

```
-- Author: JOHIR
-- Create date: 01/12/2012
                                GENERATE DATA DICTIONARY FROM SOL SERVER
-- Description:
CREATE proc [dbo].[spGenerateDBDictionary]
BEGIN
select a.name [Table], b.name [Attribute], c.name [DataType], b.isnullable [Allow Nulls?], CASE WHEN d.name is null THEN 0 ELSE 1 END [PKey?],

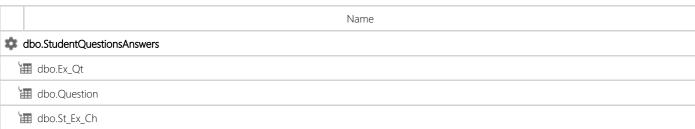
CASE WHEN e.parent_object_id is null THEN 0 ELSE 1 END [FKey?], CASE WHEN e.parent_object_id is null THEN '-' ELSE g.name END [Ref Table],

CASE WHEN h.value is null THEN '-' ELSE h.value END [Description]
from sysobjects as a
join syscolumns as b on a.id = b.id
JOIN sysobjects so ON so.id = sc.id
         JOIN sysindexkeys si ON so.id = si.id and sc.colid = si.colid
         WHERE si.indid = 1) d on a.id = d.id and b.colid = d.colid
left join sys.foreign_key_columns as e on a.id = e.parent_object_id and b.colid = e.parent_column_id
left join sys.objects as g on e.referenced_object_id = g.object_id
left join sys.extended_properties as h on a.id = h.major_id and b.colid = h.minor_id
where a.type = 'U' order by a.name
END
```

#### 2.53. Procedure: dbo.StudentQuestionsAnswers

#### Input/Output

	Name	Data type	Description
<b>→</b> @	ExamID	int	
<b>→</b> @	StuID	int	



```
CREATE PROCEDURE StudentQuestionsAnswers @ExamID int, @StuID int AS

BEGIN

select *
from St_Ex_Ch as s
inner join [dbo].[Ex_Qt] as e on s.Exam_ID = e.[Exam_ID]
inner join Question as q on q.questionID = s.questionID
```

## 2.54. Procedure: dbo.up\_generate\_data\_dictionary

#### Script

```
CREATE PROCEDURE [dbo].[up_generate_data_dictionary]
BEGIN
              Set nocount on
              DECLARE @TableName nvarchar(35)
              DECLARE Tbls CURSOR
              Select distinct Table name
              FROM INFORMATION SCHEMA.COLUMNS
               --put any exclusions here
                 -where table_name not like '%old'
              order by Table_name
              OPEN Tbls
              PRINT '<HTML><head>'
print '<style type=''text/css''>/* v1.0 | 20080212 */html, body, div, span, applet, object, iframe,hl, h2, h3, h4, h5, h6, p, blockquote, pre,a, abbr, acronym, address, big, cite, code,del, dfn, em, font, img, ins, kbd, q, s, samp,small, strike, strong, sub, sup, tt, var,b, u, i, center,dl, dt, dd, ol, ul, li,fieldset, form, label, legend,table, caption, tbody, tfoot, thead, tr, th, td { margin: 0; padding: 0; border: 0; outline: 0; font-size: 100%; vertical-align: baseline; background: transparent; lbody { line-height: 1;}ol, ul { list-style:
none;}blockquote, q { quotes: none;}blockquote:before, blockquote:after,q:before, q:after { content: ''''; content: none;}/* remember to define focus styles! */:focus { outline: 0;}/* remember to highlight inserts
                                                                                                                                 content: ''';
              /ins { text-decoration: none;}del {
print '</head><body>'
                                                                      text-decoration: line-through; } table {
somehow! */ins {
                                                                                                                               border-collapse: collapse;
              print '<h1>Data Dictionary: ' + db name() + '</h1>';
              print 'Autogenerated from database metadata at
              select GETDATE();
              print ''
              FETCH NEXT FROM Tbls
              INTO @TableName
              WHILE @@FETCH_STATUS = 0
              BEGIN
              Print '<h2>' + @TableName + '</h2>'
              PRINT ''
              --Get the Description of the table
              --Characters 1-250
Select substring(cast(Value as varchar(1000)),1,250) FROM
              sys.extended properties A
              WHERE A.major_id = OBJECT_ID(@TableName)
and name = 'MS_Description' and minor_id = 0
               --Characters 251-500
              Select substring (cast (Value as varchar (1000)), 251, 250) FROM
              sys.extended properties A
              where A.major id = OBJECT_ID(@TableName) and name = 'MS_Description' and minor_id = 0
              PRINT ''
               --Set up the Column Headers for the Table
               PRINT 'Column Name
               PRINT 'Description'
               PRINT '>InPrimaryKey
               PRINT 'IsForeignKey'
               PRINT 'DataType
               PRINT 'Length'
               PRINT 'Numeric Precision'
               PRINT 'Numeric Scale'
               PRINT 'Nullable'
               PRINT 'Computed
               PRINT '>Identity'
               PRINT 'Default Value'
              --Get the Table Data
```

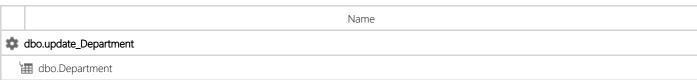
```
SELECT '',
 '<strong>' + CAST(clmns.name AS VARCHAR(35)) + '</strong>',
'\td>\t substring(ISNULL(CAST(exprop.value AS VARCHAR(255)),''),1,250),
substring(ISNULL(CAST(exprop.value AS VARCHAR(255)),''),251,250) + '
\tag{td>' + CAST(ISNULL(idxcol.index_column_id, 0)AS VARCHAR(20)) + '
\tag{td>', '\td>' + CAST(ISNULL(idxcol.index_column_id, 0)AS VARCHAR(20)) + '
\tag{td>', '\td>' + CAST(ISNULL(idxcol.index_column_id, 0)AS VARCHAR(20)) + '
\tag{td>', '\td>', '\t
 (SELECT TOP 1 1
 FROM sys.foreign_key_columns AS fkclmn
WHERE fkclmn.parent_column_id = clmns.column_id
AND fkclmn.parent_object_id = clmns.object_id
), 0) AS VARCHAR(\(\bar{2}0\)) + '
'' + CAST (udt.name AS CHAR(15)) + '' ,
'' + CAST (CAST (CASE WHEN typ.name IN (N'nchar', N'nvarchar') AND clmns.max_length <> -1
THEN clmns.max_length/2
 ELSE clmns.max_length END AS INT) AS VARCHAR(20)) + '',
 '' + CAST (CAST (clmns.precision AS INT) AS VARCHAR(20)) + '',
'' + CAST (CAST (clmns.scale AS INT) AS VARCHAR(20)) + '',
                                                                                                                                                                  '',
 '' + CAST (clmns.is_nullable AS VARCHAR(20)) + '' ,
'' + CAST (clmns.is_computed AS VARCHAR(20)) + '' ,
'' + CAST (clmns.is_identity AS VARCHAR(20)) + '' ,
 '' + isnull(CAST(cnstr.definition AS VARCHAR(20)),'') + ''''''''''''
 FROM sys.tables AS tbl
 INNER JOIN sys.all_columns AS clmns
ON clmns.object_id=tbl.object_id
LEFT OUTER JOIN sys.indexes AS idx
ON idx.object_id = clmns.object_id
AND 1 =idx.is_primary_key
LEFT OUTER JOIN sys.index_columns AS idxcol
ON idxcol.index id = idx.index id
AND idxcol.column_id = clmns.column_id
AND idxcol.object_id = clmns.object_id
AND 0 = idxcol.is_included_column
LEFT OUTER JOIN sys.types AS udt
ON udt.user_type_id = clmns.user_type_id
LEFT OUTER JOIN sys.types AS typ
ON typ.user_type_id = clmns.system_type_id
AND typ.user_type_id = typ.system_type_id
LEFT_JOIN sys.default_constraints_AS cnstr
ON cnstr.object_id=clmns.default_object_id
LEFT OUTER JOIN sys.extended_properties exprop
ON exprop.major_id = clmns.object_id
AND exprop.name = 'MS_Description'
WHERE (tbl.name = @TableName and
exprop.class = 1) --I don't want to include comments on indexes
ORDER BY clmns.column_id ASC
PRINT ''
FETCH NEXT FROM Tbls
INTO @TableName
END
PRINT '</body></HTML>'
CLOSE Tbls
DEALLOCATE Thls
```

END

#### 2.55. Procedure: dbo.update Department

#### Input/Output

	Name	Data type	Description
<b>→</b> @	DeptID	int	
<b>→</b> @	DeptName	nvarchar(20)	
<b>→</b> @	DeptDesc	nvarchar(50)	
→@	InstructorID	int	



```
CREATE PROC update_Department @DeptID int, @DeptName nvarchar(20), @DeptDesc nvarchar(50), @InstructorID int

AS

BEGIN TRY

UPDATE Department
SET depName = @DeptName,
depDescription = @DeptDesc,
ins_ID = @InstructorID
WHERE depID = @DeptID

END TRY
BEGIN CATCH
SELECT 'Failed To Update!'
END CATCH
```

# 2.56. Procedure: dbo.update\_Std\_Crs

# Input/Output

	Name	Data type	Description
<b>→</b> @	StudentID	int	
<b>→</b> @	CourseID	int	
<b>→</b> @	grade	float	

#### Uses



### Script

```
CREATE PROC update_Std_Crs @StudentID int, @CourseID int, @grade float
AS

BEGIN TRY

UPDATE Std_Crs
SET grade = @grade
WHERE std_ID = @StudentID AND crs_ID = @CourseID
END TRY
BEGIN CATCH
SELECT 'Failed To Update!'
END CATCH
```

# 2.57. Procedure: dbo.updateChoices

#### Input/Output

	Name	Data type	Description
<b>→</b> @	q_ID	int	
<b>→</b> @	choice1	nvarchar(20)	
<b>→</b> @	choice2	nvarchar(20)	
<b>→</b> @	choice3	nvarchar(20)	
→@	choice4	nvarchar(20)	



```
--UPDATE
CREATE PROC updateChoices @q_ID INT, @choice1 nvarchar(20), @choice2 nvarchar(20), @choice3 nvarchar(20), @choice4
nvarchar(20)
AS

BEGIN TRY

UPDATE Choices
SET choice1 = @choice1, choice2 = @choice2, choice3 = @choice3, choice4 = @choice4
WHERE q_ID = @q_ID
END TRY
BEGIN CATCH
SELECT 'Failed To Update!'
END CATCH
```

# 2.58. Procedure: dbo.updateCourse

# Input/Output

	Name	Data type	Description
→@	cID	int	
→@	cName	nvarchar(20)	
→@	duration	int	

#### Uses

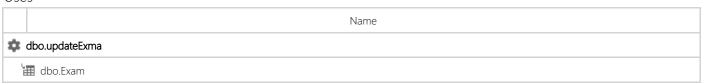


### Script

# 2.59. Procedure: dbo.updateExma

## Input/Output

	Name	Data type	Description
<b>→</b> @	Exam_ID	int	
<b>→</b> @	Exam_Date	date	



```
CREATE PROC updateExma @Exam_ID INT, @Exam_Date DATE

AS

BEGIN TRY

UPDATE Exam

SET Exam_Date = @Exam_Date

WHERE Exam_ID = @Exam_ID

END TRY

BEGIN CATCH

SELECT 'Failed to Update'

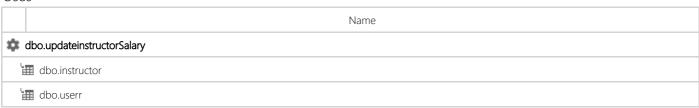
END CATCH
```

# 2.60. Procedure: dbo.updateinstructorSalary

# Input/Output

	Name	Data type	Description
<b>→</b> @	email	nvarchar(15)	
<b>→</b> @	sal	money	

#### Uses



# Script

# 2.61. Procedure: dbo.updateQuestion

# Input/Output

	Name	Data type	Description
<b>→</b> @	questionID	int	
<b>→</b> @	questionText	nvarchar(150)	
<b>→</b> @	answer	nvarchar(1)	
<b>→</b> @	qТуре	nvarchar(3)	

	Name	
🌣 d	dbo.updateQuestion	
H	₫ dbo.Question	

```
-- UPDATE
CREATE PROC updateQuestion @questionID INT, @questionText nvarchar(150), @answer nvarchar(1), @qType nvarchar(3)

AS

BEGIN TRY

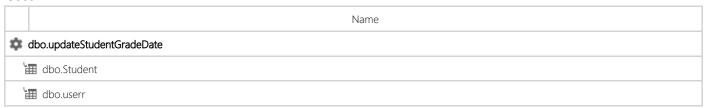
UPDATE Question
SET questionText = @questionText, answer = @answer, qType = @qType
WHERE questionID = @questionID
END TRY
BEGIN CATCH
SELECT 'Failed To Update!'
END CATCH
```

# 2.62. Procedure: dbo.updateStudentGradeDate

# Input/Output

	Name	Data type	Description
→@	email	nvarchar(15)	
<b>→</b> @	newDate	date	

#### Uses

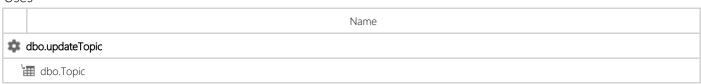


## Script

# 2.63. Procedure: dbo.updateTopic

# Input/Output

	Name	Data type	Description
→@	topic_ID	int	
→@	topic_Name	nvarchar(20)	
<b>→</b> @	courseID	int	

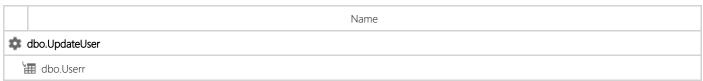


# 2.64. Procedure: dbo.UpdateUser

# Input/Output

	Name	Data type	Description
<b>→</b> @	id	int	
<b>→</b> @	firstName	nvarchar(20)	
→@	lastName	nvarchar(20)	
<b>→</b> @	email	nvarchar(15)	
<b>→</b> @	userName	nvarchar(15)	
<b>→</b> @	passKey	nvarchar(15)	
<b>→</b> @	depID	int	

#### Uses



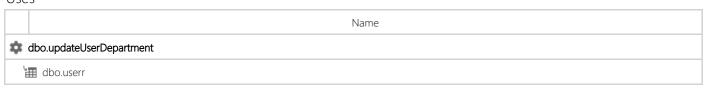
## Script

# 2.65. Procedure: dbo.updateUserDepartment

## Input/Output

	Name	Data type	Description
<b>→</b> @	email	nvarchar(15)	
<b>→</b> @	newDepId	int	

## Uses



## Script

```
CREATE proc [dbo].[updateUserDepartment] @email nvarchar(15),@newDepId int as BEGIN TRY

update userr set depID=@newDepId where email=@email

END TRY

BEGIN CATCH

SELECT 'Falied To update'

END CATCH
```