

irDevelopers.com [irDevelopers.com]

Examination System

Data Dictionary






















2/24/2022

Table of contents

Examination System	7
1. Tables	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.Ins_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. Procedures	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. Procedure: dbo.GETCourseTopics	30
2.23. Procedure: dbo.getExamQuestions	30
2.24. Procedure: dbo.getGradesWithID	31
2.25. Procedure: dbo.GETInstructorCourse	31
2.26. Procedure: dbo.GetInstructorDetails	32
2.27. Procedure: dbo.GetQuestionsWithChoices	32
2.28. Procedure: dbo.GetStudentDEtails	33
2.29. Procedure: dbo.GetStudentDtails	33

2.30.	Procedure: dbo.GETStudentGrades	33
2.31.	Procedure: dbo.GETStudentInDep	34
2.32.	Procedure: dbo.GetTFQuestionForCourse	34
2.33.	Procedure: dbo.GetUserID	34
2.34.	Procedure: dbo.GetUserIDAndType	35
2.35.	Procedure: dbo.insert_Department	35
2.36.	Procedure: dbo.insert_Std_Crs	36
2.37.	Procedure: dbo.InsertChoices	36
2.38.	Procedure: dbo.insertCourse	37
2.39.	Procedure: dbo.insertCourseWithInstrucot	37
2.40.	Procedure: dbo.insertCourseWithInstructor	38
2.41.	Procedure: dbo.insertExam	38
2.42.	Procedure: dbo.insertIns_Crs	38
2.43.	Procedure: dbo.insertInstructor	39
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	46
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
-  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 System

1. Tables

1.1. Table: dbo.Choices


Columns

		Name	Data type	Description / Attributes
		q_ID	int	References: dbo.Question
		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
	choice1, q_ID	PK__Choices__6C6330E59D1B84A2

Uses






	Name
	dbo.Choices
	dbo.Question

Used By

	Name
	dbo.Choices
	dbo.deleteChoices
	dbo.getExamQuestions
	dbo.GetQuestionsWithChoices
	dbo.InsertChoices
	dbo.updateChoices

1.2. Table: dbo.Course

Columns

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

Linked from

	Table	Join	Title / Name / Description
←	dbo.Exam	dbo.Course.cID = dbo.Exam.cID	FK_Exam_Course
←	dbo.Ins_Crs	dbo.Course.cID = dbo.Ins_Crs.crs_ID	FK__Ins_Crs__crs_ID__4BAC3F29
←	dbo.Question	dbo.Course.cID = dbo.Question.cID	FK_Question_Course
←	dbo.Std_Crs	dbo.Course.cID = dbo.Std_Crs.crs_ID	FK__Std_Crs__crs_ID__4F7CD00D
←	dbo.Topic	dbo.Course.cID = dbo.Topic.crs_ID	FK__Topic__crs_ID__440B1D61

Unique keys







	Columns	Name / Description
	cID	PK__Course__D830D457A3417035
	cName	UQ__Course__829A49F6664D5A38

Used By

	Name
	dbo.Course
	dbo.deleteCourse
	dbo.GetAviExams
	dbo.GetCoursesOfInstructor
	dbo.GETCourseTopics
	dbo.getGradesWithID
	dbo.GETInstructorCourse
	dbo.GETStudentGrades
	dbo.insertCourse
	dbo.updateCourse
←	dbo.Exam
←	dbo.Ins_Crs
←	dbo.Question
←	dbo.Std_Crs
←	dbo.Topic

1.3. Table: dbo.Department

Columns

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

Links to

	Table	Join	Title / Name / Description
➤	dbo.Instructor	dbo.Department .ins_ID = dbo.Instructor.insID	c4

Linked from

	Table	Join	Title / Name / Description
➤	dbo.Userr	dbo.Department .deplD = dbo.Userr.deplD	FK_Userr_Department

Unique keys

	Columns	Name / Description
🔑	deplD	PK_Departme__00D7A29357809BC9
🔑	depName	UQ_Departme__49814543354FF10E

Uses

	Name
📊	dbo.Department
➤	dbo.Instructor

Used By

	Name
📊	dbo.Department
⚙️	dbo.delete_Department
⚙️	dbo.insert_Department
⚙️	dbo.select_Department
⚙️	dbo.update_Department
➤	dbo.Userr

1.4. Table: dbo.Ex_Qt

Columns

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




Links to

	Table	Join	Title / Name / Description
➤	dbo.Exam	dbo.Ex_Qt .Exam_ID = dbo.Exam.Exam_ID	FK_Ex_Qt_Exam
➤	dbo.Question	dbo.Ex_Qt .questionID = dbo.Question.questionID	FK_Ex_Qt_Question






Unique keys

	Columns	Name / Description
	Exam_ID, questionID	PK_Ex_Qt

Uses





	Name
 dbo.Ex_Qt	
	dbo.Exam
	dbo.Question

Used By


	Name
 dbo.Ex_Qt	
	dbo.Exam_Answer
	dbo.GenerateExam
	dbo.getExamQuestions
	dbo.StudentQuestionsAnswers

1.5. Table: dbo.Exam


Columns

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

Links to

	Table	Join	Title / Name / Description
	dbo.Course	dbo.Exam.clD = dbo.Course.clD	FK_Exam_Course

Linked from

	Table	Join	Title / Name / Description
	dbo.Ex_Qt	dbo.Exam.Exam_ID = dbo.Ex_Qt.Exam_ID	FK_Ex_Qt_Exam
	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
	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
➤	dbo.St_Ex_Ch

1.6. Table: dbo.Ins_Crs


Columns

		Name	Data type	Description / Attributes
		ins_ID	int	References: dbo.Instructor
		crs_ID	int	References: dbo.Course


Links to

	Table	Join	Title / Name / Description
➤	dbo.Course	dbo.Ins_Crs .crs_ID = dbo.Course.clID	FK__Ins_Crs__crs_ID__4BAC3F29
➤	dbo.Instructor	dbo.Ins_Crs .ins_ID = dbo.Instructor.insID	FK__Ins_Crs__ins_ID__4AB81AF0







Unique keys

	Columns	Name / Description
	ins_ID, crs_ID	PK__Ins_Crs__527ACC3FE2DE0147

Uses




	Name
 dbo.Ins_Crs	
➤	dbo.Course
➤	dbo.Instructor

Used By

	Name
 dbo.Ins_Crs	
 dbo.AddRelationInstructorToCourse	
 dbo.GetAllInstructorsForCourse	
 dbo.GetCoursesOfInstructor	
 dbo.GETInstructorCourse	
 dbo.insertIns_Crs	

1.7. Table: dbo.Instructor



Columns

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


Links to

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

Linked from

	Table	Join	Title / Name / Description
	dbo.Department	dbo.Instructor.insID = dbo.Department.ins_ID	c4
	dbo.Ins_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.Instructor	
 dbo.Userr	







Used By

	Name
 dbo.Instructor	
 dbo.GetAllInstructorsForCourse	
 dbo.GetCoursesOfInstructor	
 dbo.GETInstructorCourse	
 dbo.GetInstructorDetails	
 dbo.insertInstructor	


	Name
	dbo.updateinstructorSalary
	dbo.Department
	dbo.Ins_Crs

1.8. Table: dbo.Question




Columns

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


Links to

	Table	Join	Title / Name / Description
	dbo.Course	dbo.Question.cID = dbo.Course.cID	FK_Question_Course

Linked from

	Table	Join	Title / Name / Description
	dbo.Choices	dbo.Question.questionID = dbo.Choices.q_ID	FK_Choices__q_ID__5441852A
	dbo.Ex_Qt	dbo.Question.questionID = dbo.Ex_Qt.questionID	FK_Ex_Qt_Question
	dbo.St_Ex_Ch	dbo.Question.questionID = dbo.St_Ex_Ch.questionID	FK_St_Ex_Ch_questi__5AEE82B9






Unique keys










	Columns	Name / Description
	questionID	PK_Question__6238D492847A7F74

Uses

	Name
	dbo.Question
	dbo.Course








Used By

	Name
	dbo.Question
	dbo.CorrectExam
	dbo.deleteQuestion
	dbo.GenerateExam
	dbo.getExamQuestions




	Name
	dbo.GetQuestionsWithChoices
	dbo.GetTFQuestionForCourse
	dbo.InsertChoices
	dbo.insertQuestion
	dbo.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
		Std_ID	int	References: dbo.Student
		Exam_ID	int	References: dbo.Exam
		questionID	int	References: dbo.Question
		St_Answer	nvarchar(1)	Nullable





Links to

	Table	Join	Title / Name / Description
	dbo.Exam	dbo.St_Ex_Ch.Exam_ID = dbo.Exam.Exam_ID	FK__St_Ex_Ch__Exam_I__59FA5E80
	dbo.Question	dbo.St_Ex_Ch.questionID = dbo.Question.questionID	FK__St_Ex_Ch__questi__5AEE82B9
	dbo.Student	dbo.St_Ex_Ch.Std_ID = dbo.Student.stuID	FK__St_Ex_Ch__Std_ID__59063A47


Unique keys






	Columns	Name / Description
	Std_ID, Exam_ID, questionID	PK__St_Ex_Ch__E13150DD47DE2FC6

Uses

	Name
	dbo.St_Ex_Ch
	dbo.Exam
	dbo.Question
	dbo.Student






Used By

	Name
	dbo.St_Ex_Ch
	dbo.CorrectExam



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

1.10. Table: dbo.Std_Crs


Columns

		Name	Data type	Description / Attributes
		std_ID	int	References: dbo.Student
		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
	dbo.Student	dbo.Std_Crs.std_ID = dbo.Student.stuID	FK__Std_Crs__std_ID__4E88ABD4











Unique keys


	Columns	Name / Description
	std_ID, crs_ID	PK__Std_Crs__C5E735C5E9C85E8B

Uses

	Name
	dbo.Std_Crs
	dbo.Course
	dbo.Student




Used By

	Name
	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
		stuID	int	References: dbo.Userr
		gradYear	date	Nullable


Links to

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

Linked from

	Table	Join	Title / Name / Description
	dbo.St_Ex_Ch	dbo.Student .stuID = dbo.St_Ex_Ch.Std_ID	FK__St_Ex_Ch__Std_ID__59063A47
	dbo.Std_Crs	dbo.Student .stuID = dbo.Std_Crs.std_ID	FK__Std_Crs__std_ID__4E88ABD4










Unique keys

	Columns	Name / Description
	stuID	PK__Student__AEC9BFAF96711633

Uses





	Name
	dbo.Student
	dbo.Userr

Used By

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

1.12. Table: dbo.Topic


Columns

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


Links to

	Table	Join	Title / Name / Description
	dbo.Course	dbo.Topic.crs_ID = dbo.Course.cID	FK__Topic__crs_ID__440B1D61






Unique keys

	Columns	Name / Description
	topicID	PK__Topic__72C15B21DEDA84B0

Uses













	Name
	dbo.Topic
	dbo.Course

Used By

	Name
	dbo.Topic
	dbo.deleteTopic
	dbo.GETCourseTopics
	dbo.insertTopic
	dbo.updateTopic

1.13. Table: dbo.Userr

Columns

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

Links to

	Table	Join	Title / Name / Description
➤	dbo.Department	dbo.Usererr.deplD = dbo.Department.deplD	FK_Usererr_Department

Linked from

	Table	Join	Title / Name / Description
➤	dbo.Instructor	dbo.Usererr.id = dbo.Instructor.insID	c3
➤	dbo.Student	dbo.Usererr.id = dbo.Student.stuID	c2

Unique keys

	Columns	Name / Description
🔑	id	PK__Usererr__3213E83FEB6608C1
🔑	userName	UQ__Usererr__66DCF95CFCD60876
🔑	email	UQ__Usererr__AB6E616405BFD6A9

Uses

	Name
📊	dbo.Usererr
➤	dbo.Department

Used By

	Name
📊	dbo.Usererr
⚙️	dbo.CheckCredential
⚙️	dbo.deleteUserByEmail
⚙️	dbo.deleteUserByuserName
⚙️	dbo.GetAllInstructorsForCourse
⚙️	dbo.GetAllStudentsForCourse
⚙️	dbo.GetInstructorDetails
⚙️	dbo.GetStudentDEtails
⚙️	dbo.GetStudentDtails
⚙️	dbo.GETStudentInDep
⚙️	dbo.GetUserID
⚙️	dbo.GetUserIDAndType
⚙️	dbo.insertInstructor
⚙️	dbo.insertStudent
⚙️	dbo.insertUser
⚙️	dbo.updateinstructorSalary
⚙️	dbo.updateStudentGradeDate
⚙️	dbo.UpdateUser
⚙️	dbo.updateUserDepartment

	Name
←	dbo.Instructor
←	dbo.Student

2. Procedures

2.1. Procedure: dbo.AddRelationInstructorToCourse

Input/Output

	Name	Data type	Description
→@	CID	int	
→@	InsID	int	
→@	STATE	int	

Uses

	Name
⚙	dbo.AddRelationInstructorToCourse
📊	dbo.Ins_Crs

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
end
```

2.2. Procedure: dbo.AddRelationStudentToCourse

Input/Output

	Name	Data type	Description
→@	CID	int	
→@	StdID	int	
→@	STATE	int	

Uses

	Name
⚙	dbo.AddRelationStudentToCourse
📊	dbo.Std_Crs

Script

```
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
```

2.3. Procedure: dbo.CheckCredential

Input/Output

	Name	Data type	Description
➔@	userName	nvarchar(15)	
➔@	passKey	nvarchar(15)	
➔@➡	isAuth	bit	

Uses

	Name
⚙️	dbo.CheckCredential
📊	dbo.Userrr

Script

```
CREATE PROC CheckCredential @userName nvarchar(15), @passKey nvarchar(15), @isAuth BIT OUT
AS
BEGIN
    IF EXISTS(SELECT ID FROM Userrr
    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
➔@	ExamID	int	
➔@	StuID	int	
➔@➡	GRADE	nvarchar(1)	

Uses

	Name
⚙️	dbo.CorrectExam
📊	dbo.Question
📊	dbo.St_Ex_Ch

Script

```
CREATE PROCEDURE CorrectExam @ExamID INT, @StuID INT, @GRADE nvarchar(1) OUT
As
BEGIN
    DECLARE
        @St_Answer NVARCHAR(1),
        @actual_answer NVARCHAR(1),
        @CID INT;

    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
    BEGIN
        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
→@	CID	int	

Uses

	Name
⚙	dbo.CourseHasExams
📊	dbo.Exam

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
→@	DeptID	int	

Uses

	Name
 dbo.delete_Department	
 dbo.Department	

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
→@	StudentID	int	
→@	CourseID	int	

Uses

	Name
 dbo.delete_Std_Crs	
 dbo.Std_Crs	

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
→@	questionID	int	

Uses

	Name
 dbo.deleteChoices	
 dbo.Choices	

Script

```
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

	Name
⚙️	dbo.deleteCourse
📊	dbo.Course

Script

```
create proc deleteCourse @courseID int
as
    begin try
        delete from Course
        where cID = @courseID
    end try
    begin catch
        select 'Falied To delete!'
    end catch
```

-----Topic Table-----
--Insert

2.10. Procedure: dbo.deleteExam

Input/Output

	Name	Data type	Description
→@	Exam_ID	int	

Uses

	Name
⚙️	dbo.deleteExam
📊	dbo.Exam

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

	Name
⚙	dbo.deleteQuestion
📊	dbo.Question

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

	Name
⚙	dbo.deleteStdSolveRelation
📊	dbo.St_Ex_Ch

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
→@	topic_ID	int	

Uses

	Name
⚙	dbo.deleteTopic

	Name
 dbo.Topic	

Script

```
create proc deleteTopic @topic_ID int
as
    begin try
        delete from Topic
        where topicID = @topic_ID
    end try
    begin catch
        select 'Falied To delete!'
    end catch
```

2.14. Procedure: dbo.deleteUserByEmail

Input/Output

	Name	Data type	Description
→@	email	nvarchar(15)	

Uses

	Name
 dbo.deleteUserByEmail	
 dbo.userr	

Script

```
create proc deleteUserByEmail @email nvarchar(15)
as BEGIN TRY
    delete from userr where email = @email
END TRY
BEGIN CATCH
    SELECT 'Falied To delete'
END CATCH
```

2.15. Procedure: dbo.deleteUserByuserName

Input/Output

	Name	Data type	Description
→@	userName	nvarchar(15)	

Uses

	Name
 dbo.deleteUserByuserName	
 dbo.userr	

Script

```
create proc deleteUserByuserName @userName nvarchar(15)
as BEGIN TRY
    delete from userr where userName = @userName
END TRY
BEGIN CATCH
    SELECT 'Falied To delete'
END CATCH
```

2.16. Procedure: dbo.Exam_Answer

Input/Output

	Name	Data type	Description
→@	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)	

Uses

	Name
⚙	dbo.Exam_Answer
📊	dbo.ex_qt
📊	dbo.St_Ex_Ch

Script





```
CREATE 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)
As
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 c1
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)
    else
        insert into St_Ex_Ch values (@StuID,@ExamID,@QuestionID,@ans10)
    set @count =@count+1
fetch c1 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	

Uses

	Name
 dbo.GenerateExam	
 dbo.Ex_Qt	
 dbo.Exam	
 dbo.Question	

Script

```
CREATE PROCEDURE [dbo].[GenerateExam] @CourseID int, @NumTF int, @NumMC int
AS
    IF @NumTF + @NumMC > 10
    BEGIN
        select 'error'
    END
    else
    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
➔@	CrsID	int	

Uses

	Name
⚙	dbo.GetAllInstructorsForCourse
📊	dbo.Ins_Crs
📊	dbo.Instructor
📊	dbo.Userr

Script

```
CREATE PROCEDURE GetAllInstructorsForCourse @CrS_ID 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 = @CrS_ID
        AND Instructor.insID = Ins_Crs.ins_ID
    END
```



2.19. Procedure: dbo.GetAllStudentsForCourse

Input/Output

	Name	Data type	Description
➔@	CrsID	int	

Uses

	Name
⚙	dbo.GetAllStudentsForCourse
📊	dbo.Std_Crs

	Name
	dbo.Student
	dbo.Userr

Script






```
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
→@	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
→@	ins_id	int	

Uses

	Name
	dbo.GetCoursesOfInstructor
	dbo.Course
	dbo.Ins_Crs


	Name
 dbo.Instructor	

Script




```
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
 @	CID	int	

Uses

	Name
 dbo.GETCourseTopics	
 dbo.Course	
 dbo.Topic	

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
 @	Exam_id	int	

Uses

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

Script

```
CREATE PROC GetExamQuestions @Exam_id INT
AS
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

	Name
⚙	dbo.getGradesWithID
📊	dbo.Course
📊	dbo.Std_Crs

Script

```
create proc [dbo].[getGradesWithID] @std_id int
as
begin
    select cName, grade from Std_Crs st
    inner join Course crs
    on st.crs_ID = crs.cID and st.std_ID = @std_id
end
```


2.25. Procedure: dbo.GETInstructorCourse

Input/Output

	Name	Data type	Description
→@	insID	int	

Uses

	Name
⚙	dbo.GETInstructorCourse
📊	dbo.Course
📊	dbo.Ins_Crs

	Name
	dbo.Instructor
	dbo.Std_Crs

Script



```
CREATE PROCEDURE GETInstructorCourse @insID INT
AS
BEGIN
    select c.cName ,count(c.cID) from Instructor i
    inner join Ins_Crs ic on ic.ins_ID =i.insID and i.insID = @insID
    inner join Course c on ic.crs_ID = c.cID
    inner join Std_Crs sc on c.cID = sc.crs_ID
    group by c.cName
End
```

2.26. Procedure: dbo.GetInstructorDetails

Input/Output

	Name	Data type	Description
→@	ins_id	int	

Uses

	Name
	dbo.GetInstructorDetails
	dbo.Instructor
	dbo.Userr

Script




```
create PROC [dbo].[GetInstructorDetails] @ins_id INT
AS
BEGIN
    SELECT * FROM Userr U
    INNER JOIN Instructor S
    ON U.ID = @ins_id AND U.id = S.insID
END
```

2.27. Procedure: dbo.GetQuestionsWithChoices

Input/Output

	Name	Data type	Description
→@	CId	int	

Uses

	Name
	dbo.GetQuestionsWithChoices
	dbo.Choices
	dbo.Question

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
→@	st_id	int	

Uses

	Name
⚙	dbo.GetStudentDEtails
📊	dbo.Student
📊	dbo.Userr

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
    END
```

2.29. Procedure: dbo.GetStudentDtails

Input/Output

	Name	Data type	Description
→@	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
→@	stuID	int	

Uses

	Name
⚙	dbo.GETStudentGrades

	Name
	dbo.Course
	dbo.Std_Crs
	dbo.Student

Script

```
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
	depID	int	

Uses

	Name
	dbo.GETStudentInDep
	dbo.Userr

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
	CID	int	

Uses

	Name
	dbo.GetTFQuestionForCourse
	dbo.Question

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
→@	userName	nvarchar(15)	
→@	ID	int	

Uses

	Name
⚙️	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
→@	userName	nvarchar(15)	
→@	ID	int	
→@	userType	nvarchar(1)	

Uses

	Name
⚙️	dbo.GetUserIDAndType
📊	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
→@	DeptName	nvarchar(20)	
→@	DeptDesc	nvarchar(50)	
→@	InstructorID	int	

Uses

	Name
 dbo.insert_Department	
 dbo.Department	

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
→@	StudentID	int	
→@	CourseID	int	
→@	grade	float	

Uses

	Name
 dbo.insert_Std_Crs	
 dbo.Std_Crs	

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
→@	q_ID	int	
→@	choice1	nvarchar(20)	
→@	choice2	nvarchar(20)	
→@	choice3	nvarchar(20)	
→@	choice4	nvarchar(20)	

Uses

	Name
 dbo.InsertChoices	
 dbo.Choices	
 dbo.Question	

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
→@	cName	nvarchar(20)	
→@	duration	int	

Uses

	Name
 dbo.insertCourse	
 dbo.Course	

Script

```
CREATE proc [dbo].[insertCourse] @cName nvarchar(20), @duration int
as
    begin try
        insert into Course(cName,duration)
        values (@cName, @duration)
    end try
    begin catch
        select 'Failed to Insert!'
    end catch
```

2.39. Procedure: dbo.insertCourseWithInstrucot

Input/Output

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

Script

```
create proc [dbo].[insertCourseWithInstrucot] @cName nvarchar(20), @duration int ,@Ins_ID int
as
begin try
exec insertCourse @cName ,@duration;
declare @CID int =IDENT_CURRENT ('user');
exec insertIns_Crs @CID,@Ins_ID;
end try
begin catch
select 'Failed to Insert!'
end catch
```

2.40. Procedure: dbo.insertCourseWithInstructor

Input/Output

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

Script

```
create proc [dbo].[insertCourseWithInstructor] @cName nvarchar(20), @duration int ,@Ins_ID int
as
begin try
exec insertCourse @cName ,@duration;
declare @CID int =IDENT_CURRENT ('Course');
exec insertIns_Crs @CID,@Ins_ID;
end try
begin catch
select 'Failed to Insert!'
end catch
```

2.41. Procedure: dbo.insertExam

Input/Output

	Name	Data type	Description
→@	Exam_Date	date	
→@	CID	int	

Uses

	Name
⚙	dbo.insertExam
📊	dbo.Exam

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
→@	crs_ID	int	
→@	ins_ID	int	

Uses

	Name
⚙	dbo.insertIns_Crs
📊	dbo.Ins_Crs

Script

```
create proc insertIns_Crs @crs_ID int, @ins_ID int
as BEGIN TRY
    INSERT INTO Ins_Crs
    values (@ins_ID,@crs_ID)
END TRY
BEGIN CATCH
    SELECT 'Falied To Insert'
END CATCH
```

2.43. Procedure: dbo.insertInstructor

Input/Output

	Name	Data type	Description
→@	fName	nvarchar(20)	
→@	lfName	nvarchar(20)	
→@	email	nvarchar(15)	
→@	userName	nvarchar(15)	
→@	password	nvarchar(15)	
→@	sex	nvarchar(1)	
→@	depId	int	
→@	sal	money	

Uses

	Name
⚙	dbo.insertInstructor
📊	dbo.instructor
📊	dbo.userr

Script

```
CREATE proc [dbo].[insertInstructor] @fName nvarchar(20),@lfName nvarchar(20),@email nvarchar(15),@userName nvarchar(15),@password nvarchar(15), @sex nvarchar(1),@depId int,@sal money
as BEGIN TRY
    INSERT INTO userr (firstName,lastName,email,userName,passKey,sex,userType,depID) values (@fName,@lfName,@email,@userName,@password,@sex,'i',@depId)
    insert into instructor values (IDENT_CURRENT ('userr'),@sal);
END TRY
BEGIN CATCH
    SELECT 'Falied To Insert'
END CATCH
```

2.44. Procedure: dbo.insertQuestion

Input/Output

	Name	Data type	Description
→@	questionText	nvarchar(150)	
→@	answer	nvarchar(1)	
→@	qType	nvarchar(3)	
→@	CID	int	

Uses

	Name
⚙️	dbo.insertQuestion
📊	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
→@	Std_ID	int	
→@	Exam_ID	int	
→@	questionID	int	
→@	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
→@	fName	nvarchar(20)	
→@	lfName	nvarchar(20)	
→@	email	nvarchar(15)	
→@	userName	nvarchar(15)	
→@	password	nvarchar(15)	
→@	sex	nvarchar(1)	
→@	depId	int	
→@	gradeDate	date	

Uses

	Name
 dbo.insertStudent	
 dbo.student	
 dbo.userr	

Script

```
CREATE proc [dbo].[insertStudent] @fName nvarchar(20),@lfName nvarchar(20),@email nvarchar(15),@userName nvarchar(15)
,@password nvarchar(15), @sex nvarchar(1),@depId INT,@gradeDate date
as BEGIN TRY
    INSERT INTO userr (firstName,lastName,email,userName,passKey,sex,userType
,depID) values (@fName,@lfName,@email,@userName,@password,@sex,'s',@depId)
    insert into student values (IDENT_CURRENT ('userr') ,@gradeDate);
END TRY
BEGIN CATCH
    SELECT 'Falied To Insert'
END CATCH
```

2.47. Procedure: dbo.insertTopic

Input/Output

	Name	Data type	Description
→@	topicName	nvarchar(20)	
→@	courseID	int	

Uses

	Name
 dbo.insertTopic	
 dbo.Topic	

Script

```
create proc insertTopic @topicName nvarchar(20), @courseID int
as
    begin try
        insert into Topic (topicName,crs_ID)
        values (@topicName, @courseID)
    end try
    begin catch
        select 'Failed to Insert!'
    end catch
--UPDATE
```

2.48. Procedure: dbo.insertUser

Input/Output

	Name	Data type	Description
→@	fName	nvarchar(20)	
→@	l fName	nvarchar(20)	
→@	email	nvarchar(15)	
→@	userName	nvarchar(15)	
→@	password	nvarchar(15)	
→@	sex	nvarchar(1)	

Uses

	Name
⚙	dbo.insertUser
📊	dbo.userr

Script

```
CREATE proc insertUser @fName nvarchar(20),@l fName 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,@l fName,@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
→@	DeptID	int	

Uses

	Name
⚙	dbo.select_Department
📊	dbo.Department

Script

```
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
→@	StudentID	int	
→@	CourseID	int	

Uses

	Name
⚙	dbo.select_Std_Crs
📊	dbo.Std_Crs

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	

Uses

	Name
⚙	dbo.SetDegree
📊	dbo.EXAM

Script

```
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
END
```

2.52. Procedure: dbo.spGenerateDBDictionary

Script

```
-- =====
-- Author:JOHIR
-- Create date: 01/12/2012
-- Description:      GENERATE DATA DICTIONARY FROM SQL SERVER
-- =====
CREATE proc [dbo].[spGenerateDBDictionary]
AS
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 systypes as c on b.xtype = c.xtype
left join (SELECT so.id,sc.colid,sc.name
FROM syscolumns sc
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
→@	ExamID	int	
→@	StuID	int	

Uses

	Name
⚙	dbo.StudentQuestionsAnswers
📊	dbo.Ex_Qt
📊	dbo.Question
📊	dbo.St_Ex_Ch

Script

```
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
END
```

2.54. Procedure: dbo.up_generate_data_dictionary

Script

```
CREATE PROCEDURE [dbo].[up_generate_data_dictionary]
AS
BEGIN

    Set nocount on

    DECLARE @TableName nvarchar(35)

    DECLARE Tbls CURSOR
    FOR

    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,h1, 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;}body { 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
somehow! */ins { text-decoration: none;}del { text-decoration: line-through;}table { border-collapse: collapse;
print '</head><body>'

    print '<h1>Data Dictionary: ' + db_name() + '</h1>';
    print '<p>Autogenerated from database metadata at '
    select GETDATE();
    print '</p>'

    FETCH NEXT FROM Tbls
    INTO @TableName

    WHILE @@FETCH_STATUS = 0
    BEGIN

        Print '<h2>' + @TableName + '</h2>'
        PRINT '<table>'
        --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 '<tr>'
        --Set up the Column Headers for the Table
        PRINT '<th>Column Name</th>'
        PRINT '<th>Description</th>'
        PRINT '<th>InPrimaryKey</th>'
        PRINT '<th>IsForeignKey</th>'
        PRINT '<th>DataType</th>'
        PRINT '<th>Length</th>'
        PRINT '<th>Numeric Precision</th>'
        PRINT '<th>Numeric Scale</th>'
        PRINT '<th>Nullable</th>'
        PRINT '<th>Computed</th>'
        PRINT '<th>Identity</th>'
        PRINT '<th>Default Value</th>'

        --Get the Table Data
```

```

SELECT '</tr>',
'<tr>',
'<td><strong>' + CAST(clmns.name AS VARCHAR(35)) + '</strong></td>',
'<td>' + substring(ISNULL(CAST(exprop.value AS VARCHAR(255)), ''), 1, 250),
substring(ISNULL(CAST(exprop.value AS VARCHAR(500)), ''), 251, 250) + '</td>',
'<td>' + CAST(ISNULL(idxcol.index_column_id, 0) AS VARCHAR(20)) + '</td>',
'<td>' + CAST(ISNULL(
(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(20)) + '</td>',
'<td>' + CAST(udt.name AS CHAR(15)) + '</td>',
'<td>' + 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)) + '</td>',
'<td>' + CAST(CAST(clmns.precision AS INT) AS VARCHAR(20)) + '</td>',
'<td>' + CAST(CAST(clmns.scale AS INT) AS VARCHAR(20)) + '</td>',
'<td>' + CAST(clmns.is_nullable AS VARCHAR(20)) + '</td>',
'<td>' + CAST(clmns.is_computed AS VARCHAR(20)) + '</td>',
'<td>' + CAST(clmns.is_identity AS VARCHAR(20)) + '</td>',
'<td>' + ISNULL(CAST(cnstr.definition AS VARCHAR(20)), '') + '</td>'
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.minor_id = clmns.column_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 '</tr></table>'

FETCH NEXT FROM Tbls
INTO @TableName
END

PRINT '</body></HTML>'

CLOSE Tbls
DEALLOCATE Tbls
END


```

2.55. Procedure: dbo.update_Department

Input/Output

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

Uses

	Name
	dbo.update_Department
	dbo.Department

Script

```
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
→@	StudentID	int	
→@	CourseID	int	
→@	grade	float	

Uses

	Name
⚙	dbo.update_Std_Crs
📊	dbo.Std_Crs

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
→@	q_ID	int	
→@	choice1	nvarchar(20)	
→@	choice2	nvarchar(20)	
→@	choice3	nvarchar(20)	
→@	choice4	nvarchar(20)	

Uses

	Name
⚙	dbo.updateChoices
📊	dbo.Choices

Script

```
--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

	Name
⚙	dbo.updateCourse
📊	dbo.Course

Script

```
CREATE proc [dbo].[updateCourse] @cID int, @cName nvarchar(20), @duration int
as
    begin try
        update Course
        set cName = @cName, duration = @duration
        where cID = @cID
    end try
    begin catch
        select 'Failed To Update!'
    end catch
-----
--DELETE
```

2.59. Procedure: dbo.updateExma

Input/Output

	Name	Data type	Description
→@	Exam_ID	int	
→@	Exam_Date	date	

Uses

	Name
⚙	dbo.updateExma
📊	dbo.Exam

Script

```
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
→@	email	nvarchar(15)	
→@	sal	money	

Uses

	Name
⚙	dbo.updateinstructorSalary
📊	dbo.instructor
📊	dbo.userr

Script

```
create proc updateinstructorSalary @email nvarchar(15),@sal money
as BEGIN TRY
    update instructor set salary=@sal from instructor i inner join userr u
    on i.insID = u.id and u.email=@email
END TRY
BEGIN CATCH
    SELECT 'Falied To update'
END CATCH
```

2.61. Procedure: dbo.updateQuestion

Input/Output

	Name	Data type	Description
→@	questionID	int	
→@	questionText	nvarchar(150)	
→@	answer	nvarchar(1)	
→@	qType	nvarchar(3)	

Uses

	Name
⚙	dbo.updateQuestion
📊	dbo.Question

Script

```
-- 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)	
➤@	newDate	date	

Uses

	Name
⚙	dbo.updateStudentGradeDate
📊	dbo.Student
📊	dbo.userr

Script

```
create proc updateStudentGradeDate @email nvarchar(15),@newDate date
as BEGIN TRY
    update Student set gradYear=@newDate from student s inner join userr u
    on s.stuID = u.id and u.email=@email
END TRY
BEGIN CATCH
    SELECT 'Falied To update'
END CATCH
```

2.63. Procedure: dbo.updateTopic

Input/Output

	Name	Data type	Description
➤@	topic_ID	int	
➤@	topic_Name	nvarchar(20)	
➤@	courseID	int	

Uses

	Name
⚙	dbo.updateTopic
📊	dbo.Topic

Script

```
create proc updateTopic @topic_ID int, @topic_Name nvarchar(20), @courseID int
as
    begin try
        update Topic
        set topicName = @topic_Name, crs_ID = @courseID
        where topicID = @topic_ID
    end try
    begin catch
        select 'Failed To Update!'
    end catch
--DELETE
```

2.64. Procedure: dbo.UpdateUser

Input/Output

	Name	Data type	Description
➤@	id	int	
➤@	firstName	nvarchar(20)	
➤@	lastName	nvarchar(20)	
➤@	email	nvarchar(15)	
➤@	userName	nvarchar(15)	
➤@	passKey	nvarchar(15)	
➤@	deplD	int	

Uses

	Name
⚙️	dbo.UpdateUser
📊	dbo.Userrr

Script


```
CREATE PROC [dbo].[UpdateUser] @id INT, @firstName nvarchar(20), @lastName nvarchar(20), @email nvarchar(15), @userName
nvarchar(15), @passKey nvarchar(15)
,@depID int
AS
BEGIN
    UPDATE Userrr
    SET firstName = @firstName,
        lastName = @lastName,
        email = @email,
        userName = @userName,
        passKey = @passKey,
        depID = @depID
    WHERE id = @id
END
```

2.65. Procedure: dbo.updateUserDepartment

Input/Output

	Name	Data type	Description
➤@	email	nvarchar(15)	
➤@	newDepId	int	

Uses

	Name
 dbo.updateUserDepartment	
 dbo.userr	

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
```