ITI-Exam System DB

Data Dictionary

2023-01-19

Table of contents

Legend

- Primary key
- Primary key disabled
- User-defined primary key
- Unique key
- Unique key disabled
- User-defined unique key
- Active trigger
- Disabled trigger
- Many to one relationship
- User-defined many to one relationship
- One to many relationship
- User-defined one to many relationship
- Many to many relationship
- User-defined many to many relationship
- One to one relationship
- User-defined one to one relationship
- Input
- Output
- Input/Output
- Uses dependency
- User-defined uses dependency
- Used by dependency
- User-defined used by dependency

ITI-Exam System DB

1. Tables

1.1. Table: course

Columns

Name	Data type	Description / Attributes
crs_id	int	**Identity / Auto increment**
cours_duration	varchar(50)	
course_name	varchar(50)	

Linked from

Table	Join	Title / Name / Description
Exam	**course**crs_id = Examcrs_id	FK_Exam_course
instructor_course	**course**crs_id = instructor_coursecrs_id	FK_instructor_course_course
Question	**course**crs_id = Questioncrs_id	FK_Question_course
student_course	**course**crs_id = student_coursecrs_id	FK_student_course_course
Topic	**course**crs_id = Topiccrs_id	FK_Topic_course

Unique keys

Columns	Name / Description
crs_id	PK_course

Used By

Name
course
Exam
instructor_course
Question
student_course
Topic

1.2. Table: department

Columns

		Name	Data type	Description / Attributes
		dept_id	int	**Identity / Auto increment**
		dept_name	varchar(50)	**Nullable**

Linked from

	Table	Join	Title / Name / Description
	instructor	**department**dept_id = instructordept_id	FK_instructor_department
	student	**department**dept_id = studentdept_id	FK_student_department

Unique keys

	Columns	Name / Description
	dept_id	PK_department

Used By

Name
department
instructor
student

1.3. Table: Exam

Columns

Name	Data type	Description / Attributes
Exam_id	int	**Identity / Auto increment**
exam_duration	int	
exam_date	date	
crs_id	int	**References**: course

Links to

Table	Join	Title / Name / Description
course	**Exam**crs_id = coursecrs_id	FK_Exam_course

Linked from

Table	Join	Title / Name / Description
Exam_Question	**Exam**Exam_id = Exam_QuestionExam_id	FK_Exam_Question_Exam
Take_exam	**Exam**Exam_id = Take_examExam_id	FK_Take_exam_Exam

Unique keys

Columns	Name / Description
Exam_id	PK_Exam

Uses

Name
Exam
course

Used By

Name
Exam
Exam_Question
Take_exam

1.4. Table: Exam_Question

Columns

Name	Data type	Description / Attributes
Exam_id	int	**References**: Exam
Q_id	int	**References**: Question

Links to

Table	Join	Title / Name / Description
Exam	**Exam_Question**Exam_id = ExamExam_id	FK_Exam_Question_Exam
Question	**Exam_Question**Q_id = QuestionQ_id	FK_Exam_Question_Question

Unique keys

Columns	Name / Description
Exam_id, Q_id	PK_Exam_Question

Uses

Name
Exam_Question
Exam
Question

1.5. Table: instructor

Columns

Name	Data type	Description / Attributes
ins_id	int	**Identity / Auto increment**
ins_name	varchar(50)	**Nullable**
salary	money	**Nullable**
dept_id	int	**Nullable** **References**: department

Links to

Table	Join	Title / Name / Description
department	**instructor**dept_id = departmentdept_id	FK_instructor_department

Linked from

Table	Join	Title / Name / Description
instructor_course	**instructor**ins_id = instructor_courseins_id	FK_instructor_course_instructor

Unique keys

Columns	Name / Description
ins_id	PK_instructor

Uses

Name
instructor
department

Used By

Name
instructor
instructor_course

1.6. Table: instructor_course

Columns

Name	Data type	Description / Attributes
ins_id	int	**References**: instructor
crs_id	int	**References**: course

Links to

Table	Join	Title / Name / Description
course	**instructor_course**crs_id = coursecrs_id	FK_instructor_course_course
instructor	**instructor_course**ins_id = instructorins_id	FK_instructor_course_instructor

Unique keys

Columns	Name / Description
ins_id, crs_id	PK_instructor_course

Uses

Name
instructor_course
course
instructor

1.7. Table: Question

Columns

Name	Data type	Description / Attributes
Q_id	int	**Identity / Auto increment**
Q_content	varchar(1000)	
Q_correct_answer	varchar(30)	
type	varchar(50)	
Q_mark	float	
crs_id	int	**References**: course

Links to

Table	Join	Title / Name / Description
course	**Question**crs_id = coursecrs_id	FK_Question_course

Linked from

Table	Join	Title / Name / Description
Exam_Question	**Question**Q_id = Exam_QuestionQ_id	FK_Exam_Question_Question
question_choices	**Question**Q_id = question_choicesQ_id	FK_question_choices_Question
Take_exam	**Question**Q_id = Take_examQ_id	FK_Take_exam_Question

Unique keys

Columns	Name / Description
Q_id	PK_Question

Uses

Name
Question
course

Used By

Name
Question
Exam_Question
question_choices
Take_exam

1.8. Table: question_choices

Columns

Name	Data type	Description / Attributes
Q_id	int	**References**: Question
choices	varchar(200)	

Links to

Table	Join	Title / Name / Description
Question	**question_choices**Q_id = QuestionQ_id	FK_question_choices_Question

Unique keys

Columns	Name / Description
Q_id, choices	PK_question_choices

Uses

Name
question_choices
Question

1.9. Table: student

Columns

Name	Data type	Description / Attributes
st_id	int	**Identity / Auto increment**
st_fname	varchar(50)	**Nullable**
st_lname	varchar(50)	**Nullable**
adress	varchar(50)	**Nullable**
age	int	**Nullable**
dept_id	int	**References**: department

Links to

Table	Join	Title / Name / Description
department	**student**dept_id = departmentdept_id	FK_student_department

Linked from

Table	Join	Title / Name / Description
student_course	**student**st_id = student_coursest_id	FK_student_course_student
student_network	**student**st_id = student_networkst_id	FK_student_network_student
Take_exam	**student**st_id = Take_examst_id	FK_Take_exam_student

Unique keys

Columns	Name / Description
st_id	PK_student

Uses

Name
student
department

Used By

Name
student
student_course
student_network
Take_exam

1.10. Table: student_course

Columns

Name	Data type	Description / Attributes
st_id	int	**References**: student
crs_id	int	**References**: course

Links to

Table	Join	Title / Name / Description
course	**student_course**crs_id = coursecrs_id	FK_student_course_course
student	**student_course**st_id = studentst_id	FK_student_course_student

Unique keys

Columns	Name / Description
st_id, crs_id	PK_student_course

Uses

Name
student_course
course
student

1.11. Table: student_network

Columns

Name	Data type	Description / Attributes
st_id	int	**References**: student
socialMediaAccount	varchar(50)	

Links to

Table	Join	Title / Name / Description
student	**student_network**st_id = studentst_id	FK_student_network_student

Unique keys

Columns	Name / Description
st_id, socialMediaAccount	PK_student_network

Uses

Name
student_network
student

1.12. Table: Take_exam

Columns

Name	Data type	Description / Attributes
st_id	int	**References**: student
Exam_id	int	**References**: Exam
Q_id	int	**References**: Question
std_answer	varchar(50)	**Nullable**
std_grade	float	**Nullable**

Links to

Table	Join	Title / Name / Description
Exam	**Take_exam**Exam_id = ExamExam_id	FK_Take_exam_Exam
Question	**Take_exam**Q_id = QuestionQ_id	FK_Take_exam_Question
student	**Take_exam**st_id = studentst_id	FK_Take_exam_student

Unique keys

Columns	Name / Description
st_id, Exam_id, Q_id	PK_Take_exam

Uses

Name
Take_exam
Exam
Question
student

1.13. Table: Topic

Columns

Name	Data type	Description / Attributes
topic_id	int	**Identity / Auto increment**
topic_name	varchar(50)	**Nullable**
crs_id	int	**Nullable** **References**: course

Links to

Table	Join	Title / Name / Description
course	**Topic**crs_id = coursecrs_id	FK_Topic_course

Unique keys

Columns	Name / Description
topic_id	PK_Topic

Uses

Name
Topic
course

2. Procedures

2.1. Procedure: addCourse

Input/Output

	Name	Data type	Description
→@	duration	int	
→@	name	varchar(50)	

Script

```
CREATE proc addCourse @duration int ,@name varchar(50)
as
begin try
          insert into course values(@duration, @name)
end try
begin catch
          select ' Invalid ----> you cannot add course'
end catch
```

2.2. Procedure: addDepartment

Input/Output

	Name	Data type	Description
→@	name	varchar(50)	

Script

```
CREATE proc addDepartment @name varchar(50)
as
begin try
          insert into department values(@name)
end try
begin catch
          select ' Invalid ----> you cannot add department'
end catch
```

2.3. Procedure: addExam

Input/Output

	Name	Data type	Description
→@	duration	int	
→@	date	date	
→@	crs_id	int	

Script

```
create proc addExam @duration int = null ,@date date = null, @crs_id int = null
as
begin try
    IF  @duration IS NULL OR @date IS NULL OR @crs_id IS NULL
        SELECT 'Missing Parameter'
    ELSE
    BEGIN
        insert into Exam values( @duration, @date,@crs_id)
        SELECT 'Exam Added Successfully'
    END
end try
begin catch
                        select ' Invalid ----> Course ID does not exist'
end catch
```

2.4. Procedure: addInstructorCourse

Input/Output

	Name	Data type	Description
→@	id	int	
→@	courseid	int	

Script

```
create proc addInstructorCourse @id int , @courseid int
as
begin try

 if exists ( select ins_id from instructor where ins_id =@id)
 and exists ( select crs_id from course where crs_id =@courseid)

           insert into instructor_course values(@id,@courseid)
end try
begin catch
           select ' Invalid Insertion '
end catch
```

2.5. Procedure: addNetworkInfo

Input/Output

	Name	Data type	Description
→@	id	int	
→@	network	varchar(50)	

Script

```
CREATE proc addNetworkInfo @id int ,@network varchar(50)
as
begin try
            insert into student_network values(@id,@network)
end try
begin catch
            select ' Invalid ----> This social network already exists'
end catch
```

2.6. Procedure: addQuestion

Input/Output

	Name	Data type	Description
@	q_content	varchar(50)	
@	q_correct_ans	varchar(50)	
@	type	varchar(50)	
@	q_mark	int	
@	crs_id	int	

Script

```
CREATE proc addQuestion
@q_content varchar(50), @q_correct_ans varchar(50), @type varchar(50), @q_mark int, @crs_id int
as
begin try
          insert into Question values(@q_content, @q_correct_ans, @type, @q_mark, @crs_id)
end try
begin catch
          select ' Invalid ----> you cannot add question'
end catch
```

2.7. Procedure: addQuestionChoices

Input/Output

	Name	Data type	Description
@	id	int	
@	A	varchar(200)	
@	B	varchar(200)	
@	C	varchar(200)	
@	D	varchar(200)	

Script

```
create proc addQuestionChoices
@id int,  @A varchar(200),@B varchar(200),@C varchar(200)=null,@D varchar(200) = null
as
begin try
           insert into question_choices values(@id,@A)
           insert into question_choices values(@id,@B)
           if @C is not null
           insert into question_choices values(@id,@C)
           if @D is not null
           insert into question_choices values(@id,@D)
end try
begin catch
           select 'invalid insertion'
end catch
```

2.8. Procedure: addStudent

Input/Output

	Name	Data type	Description
@	first_name	varchar(50)	
@	last_name	varchar(50)	
@	address	varchar(50)	
@	age	int	
@	dept_id	int	

Script

```
CREATE proc addStudent
@first_name varchar(50), @last_name varchar(50), @address varchar(50), @age int, @dept_id int
as
begin try
          insert into student values(@first_name, @last_name, @address, @age, @dept_id)
end try
begin catch
          select ' Invalid ----> you cannot add student'
end catch
```

2.9. Procedure: addStudentCourse

Input/Output

	Name	Data type	Description
→@	stu_id	int	
→@	crs_id	int	

Script

```
create proc addStudentCourse @stu_id int ,@crs_id int
as
begin try
            insert into [dbo].[student_course] values(@stu_id, @crs_id)
end try
begin catch
            select ' Invalid ----> you cannot add student_course'
end catch
```

2.10. Procedure: addTakeExam

Input/Output

	Name	Data type	Description
@	st_id	int	
@	ex_id	int	
@	q_id	int	
@	st_ans	varchar(50)	
@	st_grade	float	

Script

```
create proc addTakeExam @st_id int = null,@ex_id int = null,@q_id int = null,@st_ans varchar(50) = null,@st_grade float =
null
as
		IF @st_id IS NULL OR @ex_id IS NULL OR @q_id IS NULL OR @st_ans IS NULL OR @st_grade IS NULL
			SELECT 'Missing Parameter'
		ELSE
		BEGIN
			IF EXISTS(SELECT st_id, Exam_id, Q_id FROM Take_exam WHERE st_id= @st_id AND Exam_id= @ex_id AND Q_id=
@q_id )
				SELECT 'ROW Already Exists'
			ELSE
			BEGIN
				INSERT INTO Take_exam VALUES(@st_id, @ex_id, @q_id, @st_ans, @st_grade)
				SELECT 'TakeExam Added Successfully'
			END
		END
```

2.11. Procedure: addTopic

Input/Output

	Name	Data type	Description
→@	id	int	
→@	name	varchar(50)	
→@	courseid	int	

Script

```
create proc addTopic @id int ,@name varchar(50),@courseid int
as
begin try
          insert into Topic values(@id,@name,@courseid)
end try
begin catch
          select ' Invalid ----> you cannot add topic'
end catch
```

2.12. Procedure: deleteCourse

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
CREATE proc deleteCourse @id int
as
if exists(select crs_id from course  where crs_id = @id)
          delete from  course where crs_id = @id
else
          elect 'Invalid ----> Course does not exist'
```

2.13. Procedure: deleteDepartment

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
create proc deleteDepartment @id int
as
if exists(select dept_id from department  where dept_id = @id)
          delete from  department where dept_id = @id
else
          select 'Invalid ----> Department does not exist'
```

2.14. Procedure: deleteExam

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
create proc deleteExam @id int
as
if exists(select Exam_id from Exam  where Exam_id = @id)
          begin
                    delete from  Exam where Exam_id = @id
                    SELECT 'Exam Deleted Successfully';
          end
else
          select 'Invalid ----> Exam does not exist'
```

2.15. Procedure: deleteIns

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
create proc deleteIns @id int
as
if exists(select ins_id from instructor  where ins_id = @id)
          delete from  instructor where ins_id = @id
else
          select 'Invalid ----> instructor does not exist'
```

2.16. Procedure: deleteInstructorCourse

Input/Output

	Name	Data type	Description
→@	id	int	
→@	crsid	int	

Script

```
create proc deleteInstructorCourse @id int , @crsid int = null
as
            if @crsid is not null
            begin
                        if exists(select ins_id from instructor_course where ins_id = @id)
                        and exists ( select crs_id from instructor_course where crs_id=@crsid )
                                    delete from  instructor_course where
                                    ins_id = @id and crs_id = @crsid
            end
            else if exists(select ins_id from instructor_course where ins_id = @id)
                        delete from instructor_course where ins_id = @id
            else
                        select 'Invalid deletion for instructor courses '
```

2.17. Procedure: deleteNetworkInfo

Input/Output

	Name	Data type	Description
→@	id	int	
→@	network	varchar(50)	

Script

```
create proc deleteNetworkInfo @id int ,@network varchar(50)=null
as
if exists(select st_id from student_network where st_id = @id)
begin
if @network is not null
            delete from  student_network where st_id = @id and socialMediaAccount=@network
else
delete from  student_network where st_id = @id
end
else
            select 'Invalid ----> student does not exist'
```

2.18. Procedure: deleteQuestion

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
create proc deleteQuestion @id int
as
if exists(select Q_id from Question where Q_id=@id)
           delete from  Question where Q_id = @id
else
           select 'Invalid ----> Question does not exist'
```

2.19. Procedure: deleteQuestionChoices

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
create proc deleteQuestionChoices @id int
as
if exists(select q_id from question_choices  where Q_id = @id)
          delete from  question_choices where Q_id = @id
else
          select 'Invalid ----> question choices does not exist'
```

2.20. Procedure: deleteStudent

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
-- Delete
CREATE proc deleteStudent @id int
as
if exists(select st_id from student  where st_id = @id)
           delete from  student where st_id = @id
else
           select 'Invalid ----> student does not exist'
```

2.21. Procedure: deleteStudentCourse

Input/Output

	Name	Data type	Description
→@	stu_id	int	

Script

```
-- Delete
create proc deleteStudentCourse @stu_id int
as
if(        exists(select st_id from [dbo].[student_course] where st_id=@stu_id))
           delete from  [dbo].[student_course] where st_id = @stu_id
else
           select 'Invalid ----> student does not exist'
```

2.22. Procedure: deleteTakeExam

Input/Output

	Name	Data type	Description
→@	st_id	int	
→@	ex_id	int	
→@	q_id	int	

Script

```
CREATE proc deleteTakeExam @st_id int = null,@ex_id int = null,@q_id int = null
as
		IF @st_id IS NULL OR @ex_id IS NULL OR @q_id IS NULL
			SELECT 'Missing Parameter'
		ELSE
		BEGIN
			IF exists(select st_id,Exam_id,Q_id from Take_exam  where st_id= @st_id and Exam_id= @ex_id and Q_id =
@q_id)
			BEGIN
				delete from  Take_exam where st_id= @st_id and Exam_id= @ex_id and Q_id = @q_id
				SELECT 'Take Exam Deleted Successfully';
			END
			ELSE
				select 'Invalid ----> Exam does not exist'
		END
```

2.23. Procedure: deleteTopic

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
create proc deleteTopic @id int
as
if exists(select topic_id from Topic  where topic_id = @id)
          delete from  Topic where topic_id = @id
else
          select 'Invalid ----> Topic does not exist'
```

2.24. Procedure: examAnswers

Input/Output

	Name	Data type	Description
→@	exam_id	int	
→@	st_name	varchar(50)	
→@	q_1	varchar(50)	
→@	q_2	varchar(50)	
→@	q_3	varchar(50)	
→@	q_4	varchar(50)	
→@	q_5	varchar(50)	
→@	q_6	varchar(50)	
→@	q_7	varchar(50)	
→@	q_8	varchar(50)	
→@	q_9	varchar(50)	
→@	q_10	varchar(50)	

Script

```
CREATE proc examAnswers
@exam_id int, @st_name varchar(50), @q_1 varchar(50),
@q_2 varchar(50),@q_3 varchar(50),@q_4 varchar(50),@q_5 varchar(50),@q_6 varchar(50),
@q_7 varchar(50),@q_8 varchar(50),@q_9 varchar(50),@q_10 varchar(50)
as
        begin try
                declare @st_id int
                select @st_id = st_id from student where st_fname + ' ' + st_lname =@st_name

                if @st_id is not null
                begin
                        declare @t table(student_answer varchar(50))
                        insert into @t values(@q_1), (@q_2), (@q_3),(@q_4), (@q_5), (@q_6),(@q_7), (@q_8),
(@q_9),(@q_10)

                        --select Q_id from Question where crs_id = @
                        --insert into Take_exam
                        --select @st_id,@exam_id,student_answer from @t
                        insert into Take_exam
                        select st_id, eq.Exam_id,eq.Q_id, null, null from Exam_Question eq, student s
                        where Exam_id = @exam_id and st_id = @st_id


                        declare c2 cursor
                        for select t.std_answer from Take_exam t where t.Exam_id = @exam_id and t.st_id=@st_id
                        for update
                                declare @std_answer_Answer varchar(50)
                                open c2
                                fetch c2 into @std_answer_Answer
                                while @@FETCH_STATUS=0
                                begin
                                        declare c1 cursor
                                        for select student_answer from @t
                                        for read only
                                                declare @temp_Answer varchar(50)
                                                open c1
                                                fetch c1 into @temp_Answer
                                                while @@FETCH_STATUS=0
                                                begin
                                                        update Take_exam
                                                        set  std_answer=@temp_Answer  from Take_exam t
                                                        where current of c2
                                                        fetch c2 into @std_answer_Answer
                                                        fetch c1 into @temp_Answer
                                                end

                                end
                                close c1
                                deallocate c1
                        close c2
                        deallocate c2

                        select 'Exam Answered Successfully'
                end
                else
                        select 'this student does not exist'
        end try
        begin catch
                select 'invalid exam answering.. try again'
        end catch
```

2.25. Procedure: examCorrection

Input/Output

	Name	Data type	Description
→@	exam_id	int	
→@	st_name	varchar(50)	

Script

```
CREATE proc examCorrection @exam_id int, @st_name varchar(50)
as
        declare @st_id int
        select @st_id = st_id from student where st_fname + ' ' + st_lname =@st_name

        begin try
                if @st_id is not null
                begin
                        declare @t table(student_answer varchar(50))
                        insert into @t select q.Q_correct_answer from question q
                        inner join take_Exam TE on q.Q_id = TE.Q_id where TE.st_id=@st_id and TE.Exam_id=@exam_id

                        declare c2 cursor
                        for select t.std_answer from Take_exam t where t.Exam_id = @exam_id and t.st_id=@st_id
                        for update
                                declare @std_answer_Answer varchar(50)
                                open c2
                                fetch c2 into @std_answer_Answer
                                while @@FETCH_STATUS=0
                                begin
                                        declare c1 cursor
                                        for select student_answer from @t
                                        for read only
                                                declare @correct_Answer varchar(50)
                                                open c1
                                                        fetch c1 into @correct_Answer
                                                        while @@FETCH_STATUS=0
                                                        begin

        if(@correct_Answer=@std_answer_Answer)
                                                                        begin
                                                                                update Take_exam
                                                                                set std_grade=1
                                                                                where current of c2
                                                                        end
                                                                        else
                                                                        begin
                                                                                update Take_exam
                                                                                set std_grade=0
                                                                                where current of c2
                                                                        end
                                                                         fetch c2 into @std_answer_Answer
                                                                         fetch c1 into @correct_Answer
                                                        end
                                        end
                                close c1
                                deallocate c1
                                close c2
                                deallocate c2

                                execute getStudentGrades @st_id
        end
        else
                select 'this student does not exist'
        end try
        begin catch

        end catch
```

2.26. Procedure: generateExam

Input/Output

	Name	Data type	Description
@	crs_name	varchar(50)	
@	mcq_no	int	
@	tf_no	int	
@	ex_duration	int	
@	ex_date	date	

Script

```
CREATE proc generateExam
@crs_name varchar(50), @mcq_no int, @tf_no int, @ex_duration int =2 , @ex_date date =null
as
          if @ex_date is null set @ex_date =getDate()

          declare @crs_id int
          set @crs_id = (SELECT crs_id FROM course WHERE course_name = @crs_name)

          if @crs_id is not null
          begin
                    INSERT INTO Exam
                    VALUES (@ex_duration,@ex_date,@crs_id)

                    DECLARE @new_exam_id int = @@IDENTITY
                    --select @new_exam_id=max(Exam_id) from Exam

                    DECLARE @total_questions int = @mcq_no + @tf_no

                    INSERT INTO Exam_Question
                    SELECT TOP (@mcq_no) @new_exam_id, Q_id FROM Question WHERE crs_id = @crs_id and type ='mcq' ORDER BY
NEWID()

                    Insert Into Exam_Question
                    SELECT TOP (@tf_no) @new_exam_id, Q_id FROM Question WHERE crs_id = @crs_id and type = 'tf' ORDER BY
NEWID()
          end
          else
          select 'this course is not found'
```

2.27. Procedure: getCourse

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
CREATE proc getCourse @id int= null
as
if @id is null
          select * from course
else
          begin
                    if exists(select crs_id from course  where crs_id = @id)
                              select * from course where crs_id = @id
                    else
                              select 'course does not exist'
          end
```

2.28. Procedure: getDepartment

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
CREATE proc getDepartment @id int  =null
as
if @id is null
          select * from department
else
          begin
                    if exists(select dept_id from department  where dept_id = @id)
                                        select * from department where dept_id = @id
                    else
                              select 'Department does not exist'
          end
```

2.29. Procedure: getExam

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
CREATE proc getExam @id int = null
as
if @id is null
          select * from Exam
else
          begin
                    if exists(select Exam_id from Exam  where Exam_id = @id)
                              select * from Exam where Exam_id = @id
                    else
                              select 'Exam does not exist'
          end
```

2.30. Procedure: getInstructorCourse

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
-- select
create proc getInstructorCourse @id int = null
as
if @id is null
          select * from instructor_course
else
          begin
                    if exists(select ins_id from instructor where ins_id = @id)
                              select * from instructor_course where ins_id = @id
                    else
                              select ' this id is not invalid .. instructor does not exist'
          end
```

2.31. Procedure: getInstructorInfo

Input/Output

	Name	Data type	Description
→@	ins_id	int	

Script

```
create proc getInstructorInfo @ins_id int
as
select c.course_name , count(sc.st_id)
from instructor i, instructor_course ic, course c, student_course sc
where c.crs_id= sc.crs_id and i.ins_id = ic.ins_id and ic.crs_id = c.crs_id and
i.ins_id = @ins_id
group by c.course_name
```

2.32. Procedure: getNetworkInfo

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
CREATE proc getNetworkInfo @id int =null
as
if @id is null
          select * from student_network
else
          begin
                    if  exists(select SN.st_id from student_network SN ,student S where S.st_id=SN.st_id)
                    and exists ( select st_id from student_network where st_id=@id )
                              select * from student_network where st_id = @id
                    else
                    select ' student id not matched .. studet does not exists  '
          end
```

2.33. Procedure: getQuestion

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
CREATE proc getQuestion @id int=null
as
if @id is null
          select * from Question
else
          begin
          if exists(select Q_id from Question  where Q_id = @id)
                                select * from Question where Q_id = @id
          else
                    select 'Question does not exist'
          end
```

2.34. Procedure: getQuestionChoices

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
create proc getQuestionChoices @id  int = null
as
if @id is null
           select * from question_choices
else
begin
if @id is not null and @id in (select q_id from question_choices where Q_id=@id)
           select * from question_choices where Q_id=@id
else
           select 'no question aswers with the inserted id'
end
```

2.35. Procedure: getStudent

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
CREATE proc getStudent @id int = null
as
if @id is null
          select * from student
else
          begin
          if exists(select st_id from student  where st_id = @id)
                                select * from student where st_id = @id
          else
                    select 'student does not exist'
          end
```

2.36. Procedure: getStudentCourse

Input/Output

	Name	Data type	Description
→@	stu_id	int	

Script

```
create proc getStudentCourse @stu_id int = null
as
if @stu_id is null
          select * from [dbo].[student_course]
else
          begin
                    if(
                    exists(select st_id from [dbo].[student_course]
                    where st_id=@stu_id)
                    )
                              select * from [dbo].[student_course]  where st_id=@stu_id
                    else
                              select 'course does not exist'
          end
```

2.37. Procedure: getStudentGrades

Input/Output

	Name	Data type	Description
→@	st_id	int	

Script

```
create proc getStudentGrades @st_id int
as
select c.course_name, sum(t.std_grade) from Take_exam t, Exam e, course c where t.st_id = @st_id
and e.Exam_id = t.Exam_id and c.crs_id = e.crs_id
group by c.course_name
```

2.38. Procedure: getStudentsByDept

Input/Output

	Name	Data type	Description
→@	dept_id	int	

Script

```
create proc getStudentsByDept @dept_id int
as
select * from student where dept_id = @dept_id
```

2.39. Procedure: getTakeExam

Input/Output

	Name	Data type	Description
→@	st_id	int	
→@	ex_id	int	
→@	q_id	int	

Script

```
create proc getTakeExam @st_id int = null,@ex_id int = null,@q_id int = null
as
if @st_id is null and @ex_id is null and @q_id is null
          select * from Take_exam
else
          begin
                    if exists(select st_id, Exam_id, Q_id from Take_exam  where st_id = @st_id and Exam_id = @ex_id and
Q_id = @q_id)
                              select * from Take_exam  where st_id = @st_id and Exam_id = @ex_id and Q_id = @q_id
                    else
                              select 'Take Exam does not exist'
          end
```

2.40. Procedure: getTopic

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
CREATE proc getTopic @id int = null
as
if @id is null
          select * from Topic
else
          begin
                    if exists(select topic_id from Topic  where topic_id = @id)
                              select * from topic where topic_id = @id
                    else
                              select 'topic does not exist'
          end
```

2.41. Procedure: getTopics

Input/Output

	Name	Data type	Description
→@	crs_id	int	

Script

```
create proc getTopics @crs_id int
as
select t.topic_name
           from course c
           inner join Topic t
           on c.crs_id = t.crs_id
           where c.crs_id = @crs_id
```

2.42. Procedure: ins_show

Input/Output

	Name	Data type	Description
→@	ins_id	int	

Script

```
CREATE proc ins_show @ins_id int = null
as

if @ins_id is null
select * from instructor
else
begin
if @ins_id is not null and @ins_id in (select ins_id from instructor)
select * from instructor where ins_id=@ins_id
else
select 'your inserted id doesn not exist'
end
```

2.43. Procedure: insertIns

Input/Output

	Name	Data type	Description
→@	ins_name	varchar(30)	
→@	ins_salary	float	
→@	ins_dept	int	

Script

```
--insert instructor with the name and salary and department and check if you can insert or not
CREATE proc insertIns  @ins_name varchar(30), @ins_salary float , @ins_dept int
as
begin try
           insert into instructor(ins_name,salary,dept_id) values(@ins_name,@ins_salary,@ins_dept)
end try
begin catch
           select 'you entered wrong data'
end catch
```

2.44. Procedure: PrintExam

Input/Output

	Name	Data type	Description
→@	exam_id	int	

Script

```
CREATE proc PrintExam @exam_id int
as
select eq.Exam_id ,q.Q_content, q.type, qc.choices, q.Q_correct_answer from Exam e
           inner join Exam_Question eq on e.Exam_id = eq.Exam_id
           inner join Question q on eq.Q_id = q.Q_id
           inner join question_choices qc on q.Q_id = qc.Q_id
           where e.Exam_id = @exam_id
```

2.45. Procedure: Q_choShow

Input/Output

	Name	Data type	Description
→@	id	int	

Script

```
CREATE proc Q_choShow @id  int = null
as
if @id is null
           select * from question_choices
else
begin
if @id is not null and @id in (select q_id from question_choices where Q_id=@id)
           select * from question_choices where Q_id=@id
else
           select 'no question aswers with the inserted id'
end
```

2.46. Procedure: QuestionVsStudentAns

Input/Output

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

Script

```
CREATE proc QuestionVsStudentAns @exam_id int, @st_id int
as
select t.Exam_id,q.Q_content, q.type, q.Q_correct_answer, t.std_answer
from Take_exam t, question q
where t.Q_id = q.Q_id and t.Exam_id = @exam_id and t.st_id = @st_id
```

2.47. Procedure: updateCourse

Input/Output

	Name	Data type	Description
→@	id	int	
→@	duration	int	
→@	name	varchar(50)	

Script

```
CREATE proc updateCourse
@id int ,@duration int =NULL, @name varchar(50)=NULL
as
          if exists(select crs_id from course where crs_id=@id)
          begin
                    if @duration is not null
                              update course set cours_duration = @duration where crs_id=@id
                    if @name is not null
                              update course set course_name = @name where crs_id=@id
          end
          else
                    select 'course does not exist'
```

2.48. Procedure: updateDepartment

Input/Output

	Name	Data type	Description
→@	id	int	
→@	name	varchar(50)	

Script

```
CREATE proc updateDepartment @id int , @name varchar(50)
as
          if exists(select dept_id from department where dept_id=@id)
                                update department set dept_name = @name where dept_id=@id
          else
                    select 'department does not exist'
```

2.49. Procedure: updateExam

Input/Output

	Name	Data type	Description
→@	ex_id	int	
→@	duration	int	
→@	date	date	
→@	crs_id	int	

Script

```
create proc updateExam
@ex_id int,@duration int = null,@date date = null, @crs_id int = null
AS
begin try

            IF not exists(SELECT Exam_id FROM Exam WHERE Exam_id = @ex_id)
                        SELECT 'Invalid ----> Exam ID does not exist'

            if @duration is null  SET @duration=(SELECT exam_duration FROM Exam WHERE Exam_id = @ex_id)
            if @date is null SET @date=(SELECT exam_date FROM Exam WHERE Exam_id = @ex_id)
            if @crs_id is null SET @crs_id=(SELECT crs_id FROM Exam WHERE Exam_id = @ex_id)

            ELSE
            BEGIN
                        IF @duration >0 -- avoid negative value...
                        BEGIN
                                    UPDATE Exam SET
                                                            exam_duration= @duration,
                                                            exam_date=  @date,
                                                            crs_id= @crs_id
                                    WHERE Exam_id = @ex_id
                                    SELECT 'Exam Updated Successfully'
                        END
                        ELSE
                                    SELECT ' Invalid ----> Exam Duration must be bigger than 0'
            END
end try
begin catch

            if ISNUMERIC(@crs_id) = 0
                        select ' Invalid ----> Course ID'
            if not exists(SELECT crs_id FROM course WHERE crs_id = @crs_id)
                        select ' Invalid ----> Course ID does not exist'
end catch
```

2.50. Procedure: updateIns

Input/Output

	Name	Data type	Description
→@	id	int	
→@	name	varchar(50)	
→@	salary	float	
→@	dept_id	int	

Script

```
CREATE proc updateIns
@id int , @name varchar(50) = null,  @salary float=null, @dept_id int =null
as
begin try
            if exists( select ins_id from instructor where ins_id=@id)
            begin
                        if @name is not null
                                    update instructor set ins_name = @name where ins_id=@id
                        if @salary is not null
                                    update instructor set salary = @salary where ins_id=@id
                        if @dept_id is not null
                                    update instructor set dept_id = @dept_id where ins_id=@id
            end
            else
                        select 'instructor does not exist'
end try
begin catch
            select 'department doesnot exist'
end catch
```

2.51. Procedure: updateNetworkInfo

Input/Output

	Name	Data type	Description
→@	id	int	
→@	oldnetwork	varchar(50)	
→@	newnetwork	varchar(50)	

Script

```
create proc updateNetworkInfo
@id int ,@oldnetwork varchar(50),@newnetwork varchar(50)
as
          if exists(select st_id from student where st_id=@id)
          begin
                    if exists ( select socialMediaAccount from student_network where
                                 socialMediaAccount=@oldnetwork and st_id=@id)
                               update student_network set socialMediaAccount = @newnetwork
                               where st_id=@id and socialMediaAccount=@oldnetwork
          end
          else
                    select 'you can not update .. this student does not exist !'
```

2.52. Procedure: updateQuestion

Input/Output

	Name	Data type	Description
@	id	int	
@	q_content	varchar(50)	
@	q_correct_ans	varchar(50)	
@	type	varchar(50)	
@	q_mark	int	
@	crs_id	int	

cript

```
create proc updateQuestion
@id int , @q_content varchar(50) = NULL, @q_correct_ans varchar(50) = NULL,
@type varchar(50) = NULL, @q_mark int = NULL, @crs_id int = NULL
as
begin try
            if exists(select Q_id from Question where Q_id=@id)
            begin
                        if @q_content is not null
                                    update Question set Q_content = @q_content where Q_id=@id
                        if @q_correct_ans is not null
                                    update Question set Q_correct_answer = @q_correct_ans where Q_id=@id
                        if @type is not null
                                    update Question set type = @type where Q_id=@id
                        if @q_mark is not null
                                    update Question set Q_mark = @q_mark where Q_id=@id
                        if @crs_id is not null
                                    update Question set crs_id = @crs_id where Q_id=@id
            end
            else
                        select 'Question does not exist'
end try
begin catch
            select 'Course does not exist'
end catch
```

2.53. Procedure: updateQuestionChoices

Input/Output

	Name	Data type	Description
→@	id	int	
→@	A	varchar(200)	
→@	B	varchar(200)	
→@	C	varchar(200)	
→@	D	varchar(200)	

Script

```
CREATE proc updateQuestionChoices
@id int,  @A varchar(200) = null,@B varchar(200)=null,@C varchar(200)=null,@D varchar(200) = null
as
          if exists(select Q_id from question_choices where Q_id=@id)
          begin
                    if @A is not null
                              update question_choices set choices = @A where Q_id=@id and choices like '[Aa]%'
                    if @B is not null
                              update question_choices set choices = @B where Q_id=@id and choices like '[Bb]%'
                    if @C is not null
                              update question_choices set choices = @C where Q_id=@id and choices like '[Cc]%'
                    if @D is not null
                              update question_choices set choices = @D where Q_id=@id and choices like '[Dd]%'
          end
          else
                    select 'question does not exist'
```

2.54. Procedure: updateStudent

Input/Output

	Name	Data type	Description
@	id	int	
@	first_name	varchar(50)	
@	last_name	varchar(50)	
@	address	varchar(50)	
@	Age	int	
@	dept_id	int	

Script

```
--update
CREATE proc updateStudent
@id int , @first_name varchar(50)=NULL, @last_name varchar(50) = NULL,
@address varchar(50)=NULL, @age int=NULL, @dept_id int =NULL
as
begin try
            if exists(select st_id from student where st_id=@id)
            begin
                        if @first_name is not null
                                    update student set st_fname = @first_name where st_id=@id
                        if @last_name is not null
                                    update student set st_lname = @last_name where st_id=@id
                        if @address is not null
                                    update student set adress = @address where st_id=@id
                        if @age is not null
                                    update student set age = @age where st_id=@id
                        if @dept_id is not null
                                    update student set dept_id = @dept_id where st_id=@id
            end
            else
                        select 'student does not exist'
end try
begin catch
            select 'department does not exist'
end catch
```

2.55. Procedure: updateStudentCourse

Input/Output

	Name	Data type	Description
→@	stu_id	int	
→@	newCrs_id	int	
→@	oldCrs_id	int	

Script

```
create proc updateStudentCourse @stu_id int = 0 , @newCrs_id int = 0 , @oldCrs_id int = 0
as
            if (
            exists(select @newCrs_id from [dbo].[course]
                        where crs_id=@newCrs_id)
                        and
            exists(select st_id from [dbo].[student_course]
                        where st_id=@stu_id)
                        and
            exists(select @oldCrs_id from [dbo].[course]
                        where crs_id=@oldCrs_id)
                        )
            begin
                        if @stu_id != 0 and @newCrs_id !=0
                        begin
                        --updating student course (crs_id) depending on student ID and old course id
                        update [dbo].[student_course]  set crs_id=@newCrs_id  where st_id = @stu_id and  crs_id=@oldCrs_id
                        end
            end
            else
                        select 'Invalid update...'
```

2.56. Procedure: updateTakeExam

Input/Output

	Name	Data type	Description
@	st_id	int	
@	ex_id	int	
@	q_id	int	
@	st_ans	varchar(50)	
@	st_grade	float	

Script

```
create proc updateTakeExam @st_id int = null,@ex_id int = null,@q_id int = null,@st_ans varchar(50) = null,@st_grade float =
null
AS

            if @st_id is null  SET @st_id=(SELECT st_id FROM Take_exam WHERE st_id = @st_id)
            if @ex_id is null SET @ex_id=(SELECT Exam_id FROM Take_exam WHERE Exam_id = @ex_id)
            if @q_id is null SET @q_id=(SELECT Q_id FROM Take_exam WHERE Q_id = @q_id)
            if @st_ans is null  SET @st_ans=(SELECT std_answer FROM Take_exam WHERE std_answer = @st_ans)
            if @st_grade is null SET @st_grade=(SELECT std_grade FROM Take_exam WHERE std_grade = @st_grade)

            IF @st_id IS NULL OR @ex_id IS NULL OR @q_id IS NULL
                        SELECT 'First 3 Parameters must Inserted'
            ELSE
            BEGIN
                        if not exists(SELECT Exam_id FROM Exam WHERE Exam_id = @ex_id)
                                    select ' Invalid ----> Exam ID does not exist'
                        else if not exists(SELECT st_id FROM student WHERE st_id = @st_id)
                                    select ' Invalid ----> Student ID does not exist'
                        else if not exists(SELECT Q_id FROM Question WHERE Q_id = @q_id)
                                    select ' Invalid ----> Question ID does not exist'
                        else
                        begin
                                    UPDATE Take_exam SET
                                                            st_id = @st_id,
                                                            Exam_id= @ex_id,
                                                            Q_id= @q_id,
                                                            std_answer= @st_ans,
                                                            std_grade= @st_grade
                                    WHERE st_id= @st_id and Exam_id= @ex_id and Q_id = @q_id
                                    SELECT 'Take Exam Updated Successfully'
                        end

            END
```

2.57. Procedure: updateTopic

Input/Output

	Name	Data type	Description
→@	id	int	
→@	name	varchar(50)	
→@	course_id	int	

Script

```
--update
CREATE proc updateTopic
@id int ,@name varchar(50)=NULL,@course_id int = NULL
as
            begin try
                        if exists(select topic_id from Topic where topic_id = @id)
                                    begin
                                                if @name is not null
                                                            update Topic set topic_name = @name where topic_id = @id
                                                if @course_id is not null
                                                            update Topic set crs_id= @course_id where topic_id = @id
                                    end
                        else
                                    select 'Topic does not exist'
            end try
            begin catch
                        select 'course does not exist'
            end catch
```