Examination_System

Documentation

Table of contents

Examination_System@. 7

1. Tables 7

1.1. Table: Course 7

1.2. Table: Course_topic 8

1.3. Table: Departement 9

1.4. Table: Exam 10

1.5. Table: Exam_attempt 11

1.6. Table: Ins_Crs 12

1.7. Table: Instructor 13

1.8. Table: Ques_Ans 14

1.9. Table: Stud_Crs 15

1.10. Table: Student 16

2. Procedures 17

2.1. Procedure: AnswerQuestion 17

2.2. Procedure: Course_Delete 18

2.3. Procedure: Course_Insert 19

2.4. Procedure: Course_Select 20

2.5. Procedure: Course_topic_Delete 21

2.6. Procedure: Course_topic_Insert 22

2.7. Procedure: Course_topic_Select 23

2.8. Procedure: Course_topic_Update 24

2.9. Procedure: Course_Update 25

2.10. Procedure: CrsID_Topic 26

2.11. Procedure: Departement_Delete 27

2.12. Procedure: Departement_Insert 28

2.13. Procedure: Departement_Select 29

2.14. Procedure: Departement_Update 30

2.15. Procedure: DeptID_Student 31

2.16. Procedure: Exam_attempt_Delete 32

2.17. Procedure: Exam_attempt_Insert 33

2.18. Procedure: Exam_attempt_Select 34

2.19. Procedure: Exam_attempt_Update 35

2.20. Procedure: Exam_Delete 36

2.21. Procedure: Exam_Insert 37

2.22. Procedure: Exam_Select 38

2.23. Procedure: Exam_Update 39

2.24. Procedure: Exno_Questions_freeform 40

2.25. Procedure: Exno_Questions_freeform_solved 41

2.26. Procedure: GenerateExam 42

2.27. Procedure: GetCoursesNames 43

2.28. Procedure: GetExamIDsByCourse 44

2.29. Procedure: GetQuestion 45

2.30. Procedure: Grade_Exam 46

2.31. Procedure: Ins_Crs_Delete 47

2.32. Procedure: Ins_Crs_Insert 48
2.33. Procedure: Ins_Crs_Select 49
2.34. Procedure: Ins_Crs_Update 50
2.35. Procedure: InsID_Courses_NumStudents 51
2.36. Procedure: Instructor_Courses_Names 52
2.37. Procedure: Instructor_Delete 53
2.38. Procedure: Instructor_Insert 54
2.39. Procedure: Instructor_Select 55
2.40. Procedure: Instructor_Update 56
2.41. Procedure: PickRandomExam 57
2.42. Procedure: Ques_Ans_Delete 58
2.43. Procedure: Ques_Ans_Insert 59
2.44. Procedure: Ques_Ans_Select 60
2.45. Procedure: Ques_Ans_Update 61
2.46. Procedure: StID_Courses 62
2.47. Procedure: Stud_Crs_Delete 63
2.48. Procedure: Stud_Crs_Insert 64
2.49. Procedure: Stud_Crs_Select 65
2.50. Procedure: Stud_Crs_Update 66
2.51. Procedure: Student_Delete 67
2.52. Procedure: Student_Insert 68
2.53. Procedure: Student_Select 69
2.54. Procedure: Student_Taken_Exams 70
2.55. Procedure: Student_Update 71

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

Examination_System@.

1. Tables

1.1. Table: Course

Columns

Name			Data type	Description / Attributes
▤	🔑	crs_id	int	A Unique ID for each course, used for getting grades, course topics, and creating exams
▤		crs_name	nchar(50)	**Nullable**

Linked from

Table		Join	Title / Name / Description
→	Course_topic	**Course**crs_id = Course_topiccrs_id	FK_Course_topic_Course
→	Exam	**Course**crs_id = Examcrs_id	FK_Exam_Course
→	Ins_Crs	**Course**crs_id = Ins_Crscrs_id	FK_Ins_Crs_Course
→	Ques_Ans	**Course**crs_id = Ques_Anscrs_id	FK_Ques_Ans_Course

Unique keys

Columns		Name / Description
🔑	crs_id	PK_Course

Used By

Name
▦ **Course**
→ Course_topic
→ Exam
→ Ins_Crs
→ Ques_Ans

1.2. Table: Course_topic

Columns

Name			Data type	Description / Attributes
▤	🔑	crs_id	int	**References**: Course
▤	🔑	topic	nchar(50)	Topics discussed in this course

Links to

Table		Join	Title / Name / Description
⤚	Course	**Course_topic**crs_id = Coursecrs_id	FK_Course_topic_Course

Unique keys

Columns		Name / Description
🔑	crs_id, topic	PK_Course_topic

Uses

Name
▦ **Course_topic**
⤚ Course

1.3. Table: Departement

Columns

Name			Data type	Description / Attributes
▤	🔑	dept_id	int	Unique ID For each department, used for retrieving the data for students and their courses
▤		dept_name	nchar(50)	**Nullable**

Linked from

Table		Join	Title / Name / Description
⤙	Instructor	**Departement**dept_id = Instructordept_id	FK_Instructor_Departement
⤙	Student	**Departement**dept_id = Studentdept_id	FK_Student_Departement

Unique keys

Columns		Name / Description
🔑	dept_id	PK_Departement

Used By

Name
▦ **Departement**
⤙ Instructor
⤙ Student

1.4. Table: Exam

Columns

Name			Data type	Description / Attributes
▤	🔑	exam_id	int	**Identity / Auto increment**
▤		st_id	int	**Nullable** **References**: Student
▤		crs_id	int	**References**: Course

Links to

Table		Join	Title / Name / Description
≻—	Course	**Exam**crs_id = Coursecrs_id	FK_Exam_Course
≻—	Student	**Exam**st_id = Studentst_id	FK_Exam_Student

Linked from

Table		Join	Title / Name / Description
—≺	Exam_attempt	**Exam**exam_id = Exam_attemptexam_id	FK_Exam_attempt_Exam1
—≺	Stud_Crs	**Exam**exam_id = Stud_Crsexam_id	FK_Stud_Crs_Course

Unique keys

Columns		Name / Description
🔑	exam_id	PK__Exam__9C8C7BE9098CA7FC

Uses

Name
▦ **Exam**
≻— Course
≻— Student

Used By

Name
▦ **Exam**
—≺ Exam_attempt
—≺ Stud_Crs

1.5. Table: Exam_attempt

Columns

Name			Data type	Description / Attributes
▤	🔑	exam_id	int	**Identity / Auto increment** **References**: Exam
▤	🔑	ques_id	int	**References**: Ques_Ans
▤		stud_ans	nchar(1)	**Nullable**

Links to

Table		Join	Title / Name / Description
⤚	Exam	**Exam_attempt**exam_id = Examexam_id	FK_Exam_attempt_Exam1
⤚	Ques_Ans	**Exam_attempt**ques_id = Ques_Ansques_id	FK_Exam_attempt_Ques_Ans1

Unique keys

Columns		Name / Description
🔑	exam_id, ques_id	PK_Exam_attempt

Uses

Name
▦ **Exam_attempt**
⤚ Exam
⤚ Ques_Ans

1.6. Table: Ins_Crs

Columns

Name			Data type	Description / Attributes
▤	🔑	ins_id	int	**References**: Instructor
▤	🔑	crs_id	int	**References**: Course

Links to

Table		Join	Title / Name / Description
⤚	Course	**Ins_Crs**crs_id = Coursecrs_id	FK_Ins_Crs_Course
⤚	Instructor	**Ins_Crs**ins_id = Instructorins_id	FK_Ins_Crs_Instructor

Unique keys

Columns		Name / Description
🔑	ins_id, crs_id	PK_Ins_Crs

Uses

Name
▦ **Ins_Crs**
⤚ Course
⤚ Instructor

1.7. Table: Instructor

Columns

		Name	Data type	Description / Attributes
▤	🔑	ins_id	int	Unique Instructor ID for the instructor to log on to the system with and create exams, show courses taught, etc.
▤		dept_id	int	**Nullable** **References**: Departement
▤		fname	varchar(50)	**Nullable**
▤		lname	varchar(50)	**Nullable**
▤		Address	varchar(255)	**Nullable**
▤		Hire_date	date	**Nullable**

Links to

	Table	Join	Title / Name / Description
↣	Departement	**Instructor**dept_id = Departementdept_id	FK_Instructor_Departement

Linked from

	Table	Join	Title / Name / Description
↢	Ins_Crs	**Instructor**ins_id = Ins_Crsins_id	FK_Ins_Crs_Instructor

Unique keys

	Columns	Name / Description
🔑	ins_id	PK_Instructor

Uses

Name
▦ **Instructor**
↣ Departement

Used By

Name
▦ **Instructor**
↢ Ins_Crs

1.8. Table: Ques_Ans

Columns

		Name	Data type	Description / Attributes
▤	🔑	ques_id	int	Question ID for each question in the pool of questions, used to reference for exam creation, correction and grading later on.
▤		question	nchar(300)	**Question Body - Nullable**
▤		choiceA	nchar(100)	**Choice A that is presented for the student - Nullable**
▤		choiceB	nchar(100)	**Choice B that is presented for the student - Nullable**
▤		choiceC	nchar(100)	**Choice C that is presented for the student - Nullable**
▤		choiceD	nchar(100)	**Choice D that is presented for the student - Nullable**
▤		model_ans	nchar(1)	**A or B or C or D representing the correct answer** **- Nullable**
▤		crs_id	int	**Nullable** **References**: Course

Links to

	Table	Join	Title / Name / Description
→	Course	**Ques_Ans**crs_id = Coursecrs_id	FK_Ques_Ans_Course

Linked from

	Table	Join	Title / Name / Description
←	Exam_attempt	**Ques_Ans**ques_id = Exam_attemptques_id	FK_Exam_attempt_Ques_Ans1

Unique keys

	Columns	Name / Description
🔑	ques_id	PK_Ques_Ans

Uses

Name
▦ **Ques_Ans**
→ Course

Used By

Name
▦ **Ques_Ans**
← Exam_attempt

1.9. Table: Stud_Crs

Columns

Name			Data type	Description / Attributes
▤	🔑	st_id	int	**References**: Student
▤	🔑	crs_id	int	Course ID
▤		Grade	int	**Nullable**
▤		Pass_Fail	nchar(1)	**Nullable**
▤		CountExam	int	**Nullable**
▤		exam_id	int	**Nullable** **References**: Exam

Links to

Table		Join	Title / Name / Description
≻	Exam	**Stud_Crs**exam_id = Examexam_id	FK_Stud_Crs_Course
≻	Student	**Stud_Crs**st_id = Studentst_id	FK_Stud_Crs_Student

Unique keys

Columns		Name / Description
🔑	st_id, crs_id	PK_Stud_Crs

Uses

Name
▦ **Stud_Crs**
≻ Exam
≻ Student

1.10. Table: Student

Columns

Name			Data type	Description / Attributes
▤	🔑	st_id	int	Student ID that the student uses to log into the system with. Student can use it to show their details and grades of courses.
▤		fname	nchar(10)	**Nullable**
▤		lname	nchar(10)	**Nullable**
▤		Address	nchar(10)	**Nullable**
▤		gender	nchar(1)	**Nullable**
▤		dept_id	int	**Nullable** **References**: Departement

Links to

Table		Join	Title / Name / Description
≻—	Departement	**Student**dept_id = Departementdept_id	FK_Student_Departement

Linked from

Table		Join	Title / Name / Description
—≺	Exam	**Student**st_id = Examst_id	FK_Exam_Student
—≺	Stud_Crs	**Student**st_id = Stud_Crsst_id	FK_Stud_Crs_Student

Unique keys

Columns		Name / Description
🔑	st_id	PK_Student

Uses

Name
▦ **Student**
≻— Departement

Used By

Name
▦ **Student**
—≺ Exam
—≺ Stud_Crs

2. Procedures

2.1. Procedure: AnswerQuestion

Input/Output

Name		Data type	Description
→@	exam_id	int	Use Exam ID to enter the student answer for a specific question in a certain exam
→@	ques_id	int	Use Question ID to select a specific question present in an exam_attempt row
→@	answer	nchar(1)	One character that student picks as an answer

Script

```
CREATE PROCEDURE [dbo].[AnswerQuestion]
      @exam_id INT,
      @ques_id INT,
                @answer nchar(1)
AS
BEGI
N
      update Exam_attempt
                set stud_ans = @answer
                where @exam_id = exam_id
                and @ques_id = ques_id

                select @ques_id
END;
```

2.2. Procedure: Course_Delete

Input/Output

Name		Data type	Description
→@	crs_id	int	Course ID

Script

```
CREATE proc Course_Delete
                @crs_id int
as
begin
                delete from Course
      where crs_id = @crs_id;
end
```

2.3. Procedure: Course_Insert

Input/Output

Name		Data type	Description
→@	crs_id	int	Course ID
→@	crs_name	nchar(10)	Course Name if required to edit

Script

```
CREATE proc Course_Insert
		@crs_id as int,
		@crs_name nchar(10)
as
begin
	insert into Course (crs_id, crs_name) values
	(@crs_id, @crs_name);
end
```

2.4. Procedure: Course_Select

Input/Output

Name		Data type	Description
→@	crs_id	int	Selecting a course based on a Course ID

Script

```
CREATE proc Course_Select
		@crs_id as int
as
begin
		select * from
		Course
		where @crs_id = crs_id
end
```

2.5. Procedure: Course_topic_Delete

Input/Output

Name		Data type	Description
→@	crs_id	int	Delete a topic

Script

```
CREATE proc Course_topic_Delete
		@crs_id int
as
begin
		delete from Course_topic
	where crs_id = @crs_id;
end
```

2.6. Procedure: Course_topic_Insert

Input/Output

Name		Data type	Description
→@	crs_id	int	
→@	topic	nchar(50)	

Script

```
CREATE proc Course_topic_Insert
                @crs_id as int,
                @topic nchar(50)
as
begin
      insert into Course_topic (crs_id, topic) values
      (@crs_id, @topic);
end
```

2.7. Procedure: Course_topic_Select

Input/Output

	Name	Data type	Description
→@	crs_id	int	

Script

```
CREATE proc Course_topic_Select
            @crs_id as int
as
begin
            select *
            from Course_topic
            where @crs_id = crs_id
end
```

2.8. Procedure: Course_topic_Update

Input/Output

	Name	Data type	Description
→@	crs_id	int	
→@	topic	nchar(50)	

Script

```
CREATE proc Course_topic_Update
        @crs_id as int,
        @topic nchar(50)
as
begin
        update Course_topic
        set
                topic = @topic
        where @crs_id = crs_id
end
```

2.9. Procedure: Course_Update

Input/Output

	Name	Data type	Description
→@	crs_id	int	
→@	crs_name	nchar(10)	

Script

```
CREATE proc Course_Update
        @crs_id as int,
        @crs_name nchar(10)
as
begin
        update Course
        set
                crs_name=@crs_name
        where @crs_id = crs_id
end
```

2.10. Procedure: CrsID_Topic

Input/Output

	Name	Data type	Description
→@	crs_id	int	Course ID Used to access the topics in it

Script

```
CREATE proc CrsID_Topic
      @crs_id int
as
begin
      select C.crs_id as [Course ID],
                 C.crs_name as [Course Name],
                 Ct.topic as [Topic]
                 from Course C
                 join Course_topic Ct
                 on Ct.crs_id = C.crs_id where
                 @crs_id = C.crs_id
end;
```

2.11. Procedure: Departement_Delete

Input/Output

	Name	Data type	Description
→@	dept_id	int	

Script

```
CREATE proc Departement_Delete
                @dept_id int
as
begin
                delete from Departement
      where dept_id = @dept_id;
end
```

2.12. Procedure: Departement_Insert

Input/Output

Name		Data type	Description
→@	dept_id	int	
→@	dept_name	nchar(50)	

Script

```
CREATE proc Departement_Insert
                @dept_id int,
                @dept_name
                nchar(50)
as
begin
      insert into Departement (dept_id, dept_name) values
      (@dept_id, @dept_name);
end
```

2.13. Procedure: Departement_Select

Input/Output

Name		Data type	Description
→@	dept_id	int	

Script

```
CREATE proc Departement_Select
		@dept_id as int
as
begin
		select *
		from Departement
		where @dept_id = dept_id
end
```

2.14. Procedure: Departement_Update

Input/Output

Name		Data type	Description
→@	dept_id	int	
→@	dept_name	nchar(50)	

Script

```
CREATE proc Departement_Update
            @dept_id int,
            @dept_name
            nchar(50)
as
begin
            update
            Departement set
                        dept_name=@dept_name
            where @dept_id = dept_id
end
```

2.15. Procedure: DeptID_Student

Input/Output

	Name	Data type	Description
→@	dept_id	int	Department ID Used to retrieve all students belonging to it

Script

```
CREATE proc DeptID_Student
	@dept_id int
as
begin
	select * from
	Student
	where dept_id = @dept_id;
end;
```

2.16. Procedure: Exam_attempt_Delete

Input/Output

Name		Data type	Description
→@	Exam_id	int	

Script

```
CREATE proc Exam_attempt_Delete
            @Exam_id int
as
begin
            delete from Exam_attempt
            where @Exam_id =
            exam_id

    begin   if not exists (select 1 FROM Exam_attempt)
        dbcc checkident ('Exam_attempt', reseed, 0)
    en
en  d
d
```

2.17. Procedure: Exam_attempt_Insert

Input/Output

Name		Data type	Description
→@	Exam_id	int	
→@	ques_id	int	
→@	stud_ans	nchar(1)	

Script

```
CREATE proc Exam_attempt_Insert
                @Exam_id int,
                @ques_id int,
                @stud_ans nchar(1)
as
begin
      insert into Exam_attempt (Exam_id, ques_id, stud_ans)
      values (@Exam_id, @ques_id, @stud_ans);
end
```

2.18. Procedure: Exam_attempt_Select

Input/Output

	Name	Data type	Description
→@	Exam_id	int	

Script

```
CREATE proc Exam_attempt_Select
	@Exam_id int
as
begin
	select *
	from Exam_attempt
	where @Exam_id = exam_id
end
```

2.19. Procedure: Exam_attempt_Update

Input/Output

	Name	Data type	Description
→@	Exam_id	int	
→@	ques_id	int	
→@	stud_ans	nchar(1)	

Script

```
CREATE proc Exam_attempt_Update
		@Exam_id int,
		@ques_id int,
		@stud_ans nchar(1)
as
begin
		update
		Exam_attempt set
				stud_ans=@stud_ans
		where @Exam_id = exam_id
		and @ques_id = ques_id
end
```

2.20. Procedure: Exam_Delete

Input/Output

Name		Data type	Description
→@	Exam_id	int	

Script

```
CREATE proc Exam_Delete
            @Exam_id int
as
begin
            delete from Exam
            where @Exam_id = exam_id

            if not exists (select 1 FROM Exam)
    begin
        dbcc checkident ('Exam', reseed, 0)
    en
en  d
d
```

2.21. Procedure: Exam_Insert

Input/Output

Name		Data type	Description
→@	Exam_id	int	
→@	st_id	int	
→@	crs_id	int	

Script

```
CREATE proc Exam_Insert
                @Exam_id int,
                @st_id int, @crs_id
                int
as
begin
      insert into Exam (Exam_id, st_id, crs_id)
      values (@Exam_id, @st_id, @crs_id);
end
```

2.22. Procedure: Exam_Select

Input/Output

	Name	Data type	Description
→@	Exam_id	int	

Script

```
CREATE proc Exam_Select
		@Exam_id as int
as
begin
		select *
		from
		Exam
		where @Exam_id = exam_id
end
```

2.23. Procedure: Exam_Update

Input/Output

Name		Data type	Description
→@	Exam_id	int	
→@	st_id	int	
→@	crs_id	int	

Script

```
CREATE proc Exam_Update
        @Exam_id int,
        @st_id int, @crs_id
        int
as
begin
        update Exam
        set
                st_id = @st_id,
                crs_id = @crs_id
        where @Exam_id = exam_id
end
```

2.24. Procedure: Exno_Questions_freeform

Input/Output

Name		Data type	Description
→@	exam_id	int	Exam ID so the instructor can retrieve the questions picked by them in a free form fashion

Script

```sql
CREATE proc Exno_Questions_freeform
	@exam_id int
as
begin
	select row_number() over
		(order by QA.ques_id) as Q_Num,
		QA.question, QA.choiceA,
		QA.choiceB,QA.choiceC,QA.choiceD,
		EA.exam_id, C.crs_name
		from Ques_Ans QA
		join Exam_attempt EA
		on EA.ques_id = QA.ques_id
		join Course C
		on C.crs_id = QA.crs_id where
		EA.exam_id = @exam_id
end;
```

2.25. Procedure: Exno_Questions_freeform_solved

Input/Output

	Name	Data type	Description
→@	crs_id	int	Course ID so the student can look up their exam results in a specific Course
→@	st_id	int	Student ID to retrieve a specific grade of a course

Script

```
CREATE proc [dbo].[Exno_Questions_freeform_solved]
	@crs_id int,
		@st_id int
as
begin
	select top(10) row_number() over (order
		by QA.ques_id) as Q_Num,
		E.exam_id as [Exam ID],
		E.st_id as [Student ID], EA.stud_ans as
		[Student Answer], C.crs_name, QA.*,
		S.fname, S.lname, SC.Grade, SC.Pass_Fail from
		Ques_Ans QA
		join Exam_attempt EA
		on EA.ques_id = QA.ques_id
		join Exam E
		on E.exam_id = EA.exam_id
		join Course C
		on C.crs_id = QA.crs_id join
		Student S
		on S.st_id = E.st_id join
		Stud_Crs SC
		on E.exam_id = SC.exam_id
		where E.crs_id = @crs_id
		and E.st_id = @st_id order
		by E.exam_id desc
end;
```

2.26. Procedure: GenerateExam

Input/Output

	Name	Data type	Description
→@	crs_id	int	Randomly generate an exam for Course given Course ID, retrieve the picked 10 questions for it.

Script

```
CREATE PROCEDURE [dbo].[GenerateExam]
    @crs_id INT
AS
BEGI
N
    DECLARE @ExamID INT;

    -- Insert into Exam and get the auto-incremented exam_id INSERT
    INTO Exam (crs_id)
    VALUES (@crs_id);

    SET @ExamID = SCOPE_IDENTITY(); -- Get the newly generated exam_id

    -- Create a temporary table for selected questions CREATE
    TABLE #TempExam (
        Q_id INT,
        Q NVARCHAR(300),
        A NVARCHAR(100),
        B NVARCHAR(100),
        C NVARCHAR(100),
        D NVARCHAR(100)
    );

    -- Insert selected questions into temp table INSERT
    INTO #TempExam (Q_id, Q, A, B, C, D)
    SELECT ques_id, question, choiceA, choiceB, choiceC, choiceD FROM
    (
        SELECT TOP(3) ques_id, question, choiceA, choiceB, choiceC, choiceD, 1 AS Priority FROM
        Ques_Ans
        WHERE crs_id = @crs_id AND choiceC IS NULL
        ORDER BY NEWID()

        UNION ALL

        SELECT TOP(7) ques_id, question, choiceA, choiceB, choiceC, choiceD, 2 AS Priority FROM
        Ques_Ans
        WHERE crs_id = @crs_id AND choiceC IS NOT
        NULL ORDER BY NEWID()
    ) AS CombinedResults
    ORDER BY Priority, NEWID();

    -- Insert questions into Exam_attempt
            SET IDENTITY_INSERT Exam_attempt ON;

    INSERT INTO Exam_attempt (exam_id, ques_id, stud_ans)
    SELECT @ExamID, Q_ID, NULL FROM #TempExam;

            SELECT @ExamID AS ExamID;

END;
```

2.27. Procedure: GetCoursesNames

Script

```
create proc GetCoursesNames
as
select crs_name from Course
```

2.28. Procedure: GetExamIDsByCourse

Input/Output

Name		Data type	Description
→@	ins_id	int	Retrieve all Courses exams created by an instructor

Script

```
CREATE PROCEDURE
    GetExamIDsByCourse @ins_id int
AS
BEGI
N
      SET NOCOUNT ON;

      SELECT
      Exam.exam_id
      FROM Exam
      WHERE Exam.crs_id IN (select Course.crs_id from Ins_Crs inner join Course on
Course.crs_id = Ins_Crs.crs_id
where Ins_Crs.ins_id = @ins_id
)
END
;
```

2.29. Procedure: GetQuestion

Input/Output

Name		Data type	Description
→@	exam_id	int	Get all 10 questions present in an exam using its ID

Script

```
create proc GetQuestion
                @exam_id
                int as
begin
select Ques_Ans.*
from Ques_Ans
join Exam_attempt
on Exam_attempt.ques_id = Ques_Ans.ques_id
where exam_id=@exam_id
end;
```

2.30. Procedure: Grade_Exam

Input/Output

Name		Data type	Description
→@	exam_id	int	Exam ID with which a grade will be given for the student
→@	std_id	int	Student that will receive a new grade
→@	crs_id	int	Specify the course whose grade will be edited

Script

```
CREATE proc [dbo].[Grade_Exam] @exam_id int, @std_id int, @crs_id int as
DECLARE @correct_count INT;

SELECT @correct_count =
COUNT(*) FROM Exam_attempt
INNER JOIN Ques_Ans
ON Exam_attempt.ques_id = Ques_Ans.ques_id
WHERE stud_ans LIKE model_ans
AND Exam_attempt.exam_id = @exam_id;

IF @correct_count >=5
                              update Stud_Crs set Grade = @correct_count*10, Pass_Fail = 'P', exam_id = @exam_id
                               WHERE Stud_Crs.st_id = @std_id AND Stud_Crs.crs_id = @crs_id;
ELSE
    update Stud_Crs set Grade = @correct_count*10, Pass_Fail = 'F', exam_id = @exam_id WHERE
     Stud_Crs.st_id = @std_id AND Stud_Crs.crs_id = @crs_id;
select @correct_count
```

2.31. Procedure: Ins_Crs_Delete

Input/Output

	Name	Data type	Description
→@	ins_id	int	

Script

```
CREATE proc Ins_Crs_Delete
		@ins_id int
as
begin
		delete from Ins_Crs
		where @ins_id = ins_id
end
```

2.32. Procedure: Ins_Crs_Insert

Input/Output

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

Script

```
CREATE proc Ins_Crs_Insert
                @ins_id int, @crs_id
                int
as
begin
      insert into Ins_Crs (ins_id, crs_id) values
      (@ins_id, @crs_id);
end
```

2.33. Procedure: Ins_Crs_Select

Input/Output

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

Script

```
CREATE proc Ins_Crs_Select
        @ins_id int, @crs_id
        int
as
begin
        select * from
        Ins_Crs
        where @ins_id = ins_id
        and @crs_id = crs_id
end
```

2.34. Procedure: Ins_Crs_Update

Input/Output

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

Script

```
CREATE proc Ins_Crs_Update
        @ins_id int, @crs_id
        int
as
begin
        update Ins_Crs
        set
                crs_id = @crs_id
        where @ins_id = ins_id
        and @crs_id = crs_id
end
```

2.35. Procedure: InsID_Courses_NumStudents

Input/Output

Name		Data type	Description
→@	ins_id	int	For an instructor, get all students enrolled in a course they teach

Script

```
CREATE proc InsID_Courses_NumStudents
      @ins_id int
as
begin
      select I.ins_id, I.ins_name, C.crs_name,
                 count(SC.st_id) as [Number of
                 Enrolled Students] from Course C
                 join Ins_Crs IC
                 on IC.crs_id = c.crs_id join
                 Instructor I
                 on I.ins_id = IC.ins_id join
                 Stud_Crs SC
                 on SC.crs_id = c.crs_id where
                 @ins_id = I.ins_id
                 group by I.ins_id, I.ins_name, C.crs_name
end;
```

2.36. Procedure: Instructor_Courses_Names

Input/Output

	Name	Data type	Description
→@	ins_id	int	Getting course names taught by the instructor

Script

```
CREATE Proc Instructor_Courses_Names @ins_id int
as
select Course.crs_id,Course.crs_name from Ins_Crs inner join Course on
Course.crs_id = Ins_Crs.crs_id
where Ins_Crs.ins_id = @ins_id
```

2.37. Procedure: Instructor_Delete

Input/Output

	Name	Data type	Description
→@	ins_id	int	

Script

```
CREATE proc Instructor_Delete
        @ins_id int
as
begin
        delete from Instructor
        where @ins_id = ins_id
end
```

2.38. Procedure: Instructor_Insert

Input/Output

	Name	Data type	Description
→@	ins_id	int	
→@	ins_name	nchar(20)	
→@	dept_id	int	

Script

```
CREATE proc Instructor_Insert
                @ins_id int,
                @ins_name nchar(20),
                @dept_id int
as
begin
      insert into Instructor (ins_id, ins_name, dept_id) values
      (@ins_id, @ins_name, @dept_id);
end
```

2.39. Procedure: Instructor_Select

Input/Output

Name		Data type	Description
→@	ins_id	int	

Script

```
CREATE proc Instructor_Select
                @ins_id int
as
begin
                select *
                from Instructor
                where @ins_id = ins_id
end
```

2.40. Procedure: Instructor_Update

Input/Output

Name		Data type	Description
→@	ins_id	int	
→@	ins_name	nchar(20)	
→@	dept_id	int	

Script

```
CREATE proc Instructor_Update
		@ins_id int,
		@ins_name nchar(20),
		@dept_id int
as
begin
		update Instructor
		set
				ins_name = @ins_name,
				dept_id = @dept_id
		where @ins_id = ins_id
end
```

2.41. Procedure: PickRandomExam

Input/Output

	Name	Data type	Description
→@	crs_id	int	For a course, pick a random exam from a group of generated exams for the same course
→@	st_id	int	Assign the exam to a specific student using their ID

Script

```
CREATE PROCEDURE [dbo].[PickRandomExam]
    @crs_id INT,
    @st_id INT
AS
BEGI
N
            DECLARE @count INT;
    SELECT @count =
    COUNT(*) FROM Exam E
    WHERE E.crs_id = @crs_id AND E.st_id = @st_id;



    IF @count > 1
    BEGIN
                        UPDATE Stud_Crs
            set CountExam = 2
            where Stud_Crs.crs_id = @crs_id AND Stud_Crs.st_id = @st_id; PRINT
        'Student has already taken 2 or more exams for this course.'; RETURN;
    END;

    DECLARE @RandomExamID INT;
    -- Select a random exam_id from available exams for this course and student SELECT
    TOP 1 @RandomExamID = exam_id
    FROM Exam
    WHERE crs_id = @crs_id AND st_id IS NULL
    ORDER BY NEWID();  -- Random selection
    -- If no exam exists, return NULL IF
    @RandomExamID IS NULL
    BEGIN
        PRINT 'No available exams for this course.';
        RETURN;
    END

            update Exam
            set
                        st_id = @st_id where
            @RandomExamID = exam_id
            select @RandomExamID
END;
```

2.42. Procedure: Ques_Ans_Delete

Input/Output

Name	Data type	Description
→@ ques_id	int	

Script

```
CREATE proc Ques_Ans_Delete
            @ques_id int
as
begin
            delete from Ques_Ans
            where @ques_id = ques_id
end
```

2.43. Procedure: Ques_Ans_Insert

Input/Output

Name		Data type	Description
→@	ques_id	int	
→@	choiceA	nchar(100)	
→@	choiceB	nchar(100)	
→@	choiceC	nchar(100)	
→@	choiceD	nchar(100)	
→@	model_ans	nchar(1)	

Script

```
CREATE proc Ques_Ans_Insert
            @ques_id int,
            @choiceA nchar(100),
            @choiceB nchar(100),
            @choiceC nchar(100),
            @choiceD nchar(100),
            @model_ans nchar(1)
as
begin
    insert into Ques_Ans (
            ques_id, choiceA , choiceB, choiceC,
            choiceD,model_ans)
    values (
            @ques_id, @choiceA , @choiceB,
            @choiceC,@choiceD,@model_ans);
end
```

2.44. Procedure: Ques_Ans_Select

Input/Output

Name		Data type	Description
→@	ques_id	int	

Script

```
CREATE proc Ques_Ans_Select
        @ques_id int
as
begin
        select  * from
        Ques_Ans
        where @ques_id = ques_id
end
```

2.45. Procedure: Ques_Ans_Update

Input/Output

Name		Data type	Description
→@	ques_id	int	
→@	choiceA	nchar(100)	
→@	choiceB	nchar(100)	
→@	choiceC	nchar(100)	
→@	choiceD	nchar(100)	
→@	model_ans	nchar(1)	

Script

```
CREATE proc Ques_Ans_Update
		@ques_id int,
		@choiceA nchar(100),
		@choiceB nchar(100),
		@choiceC nchar(100),
		@choiceD nchar(100),
		@model_ans nchar(1)
as
begin
		update
		Ques_Ans set
				choiceA = @choiceA,
				choiceB = @choiceB,
				choiceC = @choiceC,
				choiceD = @choiceD,
				model_ans = @model_ans
		where @ques_id = ques_id
end
```

2.46. Procedure: StID_Courses

Input/Output

Name		Data type	Description
→@	st_id	int	Get all courses that a student is assigned with

Script

```
CREATE proc [dbo].[StID_Courses]
      @st_id int
as
begin
                  select CONCAT(LTRIM(RTRIM(S.fname)), ' ', LTRIM(RTRIM(S.lname))) AS Fullname, SC.st_id,
                  SC.crs_id, C.crs_name,
                  SC.Grade, SC.Pass_Fail,
                  SC.CountExam,
                  SC.exam_id from Stud_Crs
                  SC
                  join Student   S
                  on SC.st_id = S.st_id join
                  Course C
                  on C.crs_id = SC.crs_id
      where S.st_id = @st_id; end;
```

2.47. Procedure: Stud_Crs_Delete

Input/Output

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

Script

```
CREATE proc Stud_Crs_Delete
                @st_id int, @crs_id
                int
as
begin
                delete  from  Stud_Crs
                where  @st_id  =  st_id
                and @crs_id = crs_id
end
```

2.48. Procedure: Stud_Crs_Insert

Input/Output

Name		Data type	Description
→@	st_id	int	
→@	crs_id	int	
→@	Grade	int	
→@	Pass_Fail	nchar(1)	

Script

```
CREATE proc Stud_Crs_Insert
                @st_id int, @crs_id
                int, @Grade int,
                @Pass_Fail nchar(1)
as
begin
      insert into Stud_Crs (
                st_id, crs_id , Grade, Pass_Fail) values (
                @st_id, @crs_id , @Grade, @Pass_Fail);
end
```

2.49. Procedure: Stud_Crs_Select

Input/Output

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

Script

```
CREATE proc Stud_Crs_Select
		@st_id int, @crs_id
		int
as
begin
		select  * from
		Stud_Crs
		where @st_id = st_id
		and @crs_id = crs_id
end
```

2.50. Procedure: Stud_Crs_Update

Input/Output

	Name	Data type	Description
→@	st_id	int	
→@	crs_id	int	
→@	Grade	int	
→@	Pass_Fail	nchar(1)	

Script

```
CREATE proc Stud_Crs_Update
                @st_id int, @crs_id
                int, @Grade int,
                @Pass_Fail nchar(1)
as
begin
                update Stud_Crs
                set
                                Grade = @Grade,
                                Pass_Fail = @Pass_Fail
                where @st_id = st_id
                and @crs_id = crs_id
end
```

2.51. Procedure: Student_Delete

Input/Output

Name		Data type	Description
→@	st_id	int	

Script

```
CREATE proc Student_Delete
                @st_id int
as
begin
                delete from Student
        where st_id = @st_id;
end
```

2.52. Procedure: Student_Insert

Input/Output

	Name	Data type	Description
→@	st_id	int	
→@	fname	nchar(10)	
→@	lname	nchar(10)	
→@	address	nchar(10)	
→@	gender	nchar(1)	
→@	dept_id	int	

Script

```
CREATE proc Student_Insert
        @st_id int,
        @fname nchar(10),
        @lname nchar(10),
        @address nchar(10),
        @gender nchar(1),
        @dept_id int
as
begin
        insert into Student (st_id, fname, lname, address, gender, dept_id) values
        (@st_id, @fname, @lname, @address, @gender, @dept_id);
end
```

2.53. Procedure: Student_Select

Input/Output

	Name	Data type	Description
→@	st_id	int	

Script

```
CREATE proc Student_Select
		@st_id as int
as
begin
		select * from
		Student
		where @st_id = st_id
end
```

2.54. Procedure: Student_Taken_Exams

Input/Output

Name		Data type	Description
→@	st_id	int	Retrieve all exams taken by student, only the most recent ones

Script

```
CREATE proc [dbo].[Student_Taken_Exams] @st_id int
as
select distinct Exam.crs_id from Exam where Exam.st_id = @st_id
```

2.55. Procedure: Student_Update

Input/Output

Name		Data type	Description
→@	st_id	int	
→@	fname	nchar(10)	
→@	lname	nchar(10)	
→@	address	nchar(10)	
→@	gender	nchar(1)	
→@	dept_id	int	

Script

```
CREATE proc Student_Update
            @st_id as int,
            @fname nchar(10),
            @lname nchar(10),
            @address nchar(10),
            @gender nchar(1),
            @dept_id int
as
begin
            update Student
            set
                        fname = @fname,
                        lname = @lname,
                        address = @address,
                        gender = @gender,
                        dept_id = @dept_id
            where @st_id = st_id
end
```