DESKTOP-7DE5S9C\SQLEXPRESS

Examination

Server	DESKTOP-7DE5S9C\SQLEXPRESS
Author	Uccief Mhmd
Created	Wednesday, January 17, 2024 2:32:40 AM
File Path	E:\Examination Exam\Script documentaition File-2024-01-17T02-32-40.pdf

Table of Contents

DESKTOP-7DE5S9C\SQLEXPRESS

Databases (1)

- Examination

Server Properties

Property	Value
Product	Microsoft SQL Server
Version	16.0.1000.6
Language	English (United States)
Platform	NT x64
Edition	Express Edition (64-bit)
Engine Edition	4 (Express)
Processors	8
OS Version	6.3 (19045)
Physical Memory	16265
Is Clustered	False
Root Directory	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL
Collation	Arabic_CI_AS

Server Settings

Property	Value
Default data file path	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\
Default backup file path	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\Backup
Default log file path	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\
Recovery Interval (minutes)	0
Default index fill factor	0
Default backup media retention	0

Advanced Server Settings

Property	Value
Locks	0
Nested triggers enabled	True
Allow triggers to fire others	True
Default language	English
Network packet size	4096

Default fulltext language LCID	1033
Two-digit year cutoff	2049
Remote login timeout	10
Cursor threshold	-1
Max text replication size	65536
Parallelism cost threshold	5
Max degree of parallelism	0
Min server memory	16
Max server memory	2147483647
Scan for startup procs	False
Transform noise words	False
CLR enabled	False
Blocked process threshold	0
Filestream access level	False
Optimize for ad hoc workloads	False
CLR strict security	True

User databases

Databases (1)

- Examination

Examination Database

Files

Name	Type	File Group	Size	Maxsize	Autogrowth	File Name
Examination	Data		72.00 MB	unlimited	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\Examination.mdf
Examination_log	Log		8.00 MB	2048.00 GB	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\Examination_log.ldf
ExamGroup	Data	Fourth	8.00 MB	unlimited	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\ExamGroup.ndf
studentGroup	Data	Secoundary	8.00 MB	unlimited	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\studentGroup.ndf
InstructorGroup	Data	Third	8.00 MB	unlimited	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\InstructorGroup.ndf

Tables

Objects

Name
dbo.Answer
dbo.Branchs
dbo.Class
dbo.Courses
dbo.Department
dbo.Exam
dbo.Exam_Question
dbo.Inst_teach_course
dbo.Instructors
dbo.Intake
dbo.Intake_Branch
dbo.Intake_Depart
dbo.Questions
dbo.Student_Exam
dbo.Students
dbo.Tracks

[dbo].[Answer]

Properties

Property	Value
Collation	Arabic_CI_AS
Row Count (~)	6
Created	12:56:39 AM Monday, January 15, 2024
Last Modified	12:03:17 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
🔑🔑	Student_ID	int	4	NOT NULL
🔑🔑	Exam_ID	int	4	NOT NULL
🔑🔑	Question_ID	int	4	NOT NULL
	Student_answer	nvarchar(50)	100	NOT NULL

Indexes

Key	Name	Key Columns	Unique
🔑	PK_Student_Exam_Ques	Student_ID, Exam_ID, Question_ID	True

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
UpdateResultsTrigger	True	True	After Insert

Foreign Keys

Name	Update	Delete	Columns
FK_Student_Exam_Ques_Exam	Cascade	Cascade	Exam_ID->[dbo].[Exam].[E_ID]
FK_Student_Exam_Ques_Questions			Question_ID->[dbo].[Questions].[Q_ID]
FK_Student_Exam_Ques_Students			Student_ID->[dbo].[Students].[Std_ID]

SQL Script

```
CREATE TABLE [dbo].[Answer]
```

```
(
[Student_ID] [int] NOT NULL,
[Exam_ID] [int] NOT NULL,
[Question_ID] [int] NOT NULL,
[Student_answer] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL
) ON [PRIMARY]
GO
CREATE TRIGGER [dbo].[UpdateResultsTrigger]
ON [dbo].[Answer]
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StudentID INT, @ExamID INT;

    -- Get the Student_ID and Exam_ID from the inserted rows
    SELECT @StudentID = Student_ID, @ExamID = Exam_ID
    FROM inserted;

    -- Update the results using the stored procedure
    EXEC dbo.UpdateStudentExamResults @StudentID, @ExamID;
END;
GO
ALTER TABLE [dbo].[Answer] ADD CONSTRAINT [PK_Student_Exam_Ques] PRIMARY KEY CLUSTERED
([Student_ID], [Exam_ID], [Question_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Answer] ADD CONSTRAINT [FK_Student_Exam_Ques_Exam] FOREIGN KEY ([Exam_ID])
REFERENCES [dbo].[Exam] ([E_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[Answer] ADD CONSTRAINT [FK_Student_Exam_Ques_Questions] FOREIGN KEY
([Question_ID]) REFERENCES [dbo].[Questions] ([Q_ID])
GO
ALTER TABLE [dbo].[Answer] ADD CONSTRAINT [FK_Student_Exam_Ques_Students] FOREIGN KEY ([Student_-
ID]) REFERENCES [dbo].[Students] ([Std_ID])
GO
```

Uses

[dbo].[Exam]
[dbo].[Questions]
[dbo].[Students]

Used By

[StudentSC].[InsertAnswer]
[dbo].[CalculateTotalCorrectAnswers]

[dbo].[Branchs]

Properties

Property	Value
Collation	Arabic_CI_AS
File Group	Secoundary
Row Count (~)	11
Created	1:10:48 AM Monday, January 15, 2024
Last Modified	12:38:43 AM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Branch_ID	int	4	NOT NULL	1 - 1
	Branch_name	nvarchar(50)	100	NOT NULL	
	Branch_address	nvarchar(50)	100	NOT NULL	

Indexes

Key	Name	Key Columns	Unique	File Group
PK C	PK_Branchs	Branch_ID	True	Secoundary

SQL Script

```
CREATE TABLE [dbo].[Branchs]
(
[Branch_ID] [int] NOT NULL IDENTITY(1, 1),
[Branch_name] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[Branch_address] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL
) ON [Secoundary]
GO
ALTER TABLE [dbo].[Branchs] ADD CONSTRAINT [PK_Branchs] PRIMARY KEY CLUSTERED ([Branch_ID]) ON
[Secoundary]
GO
```

Used By

[dbo].[Exam]
[dbo].[Intake_Branch]
[dbo].[Students]

[InstructorSC].[ShowAllDataFromExam]
[MangerSC].[ShowAllDataFromBranch]
[MangerSC].[ShowAllDataFromStudent]
[MangerSC].[CreateBranch]
[MangerSC].[EidtBranch]

[dbo].[Class]

Properties

Property	Value
Collation	Arabic_CI_AS
File Group	Fourth
Row Count (~)	8
Created	1:11:30 AM Monday, January 15, 2024
Last Modified	12:36:16 AM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Class_ID	int	4	NOT NULL	1 - 1
	Class_name	nvarchar(50)	100	NOT NULL	

Indexes

Key	Name	Key Columns	Unique	File Group
PK C	PK_Class	Class_ID	True	Fourth

SQL Script

```
CREATE TABLE [dbo].[Class]
(
[Class_ID] [int] NOT NULL IDENTITY(1, 1),
[Class_name] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL
) ON [Fourth]
GO
ALTER TABLE [dbo].[Class] ADD CONSTRAINT [PK_Class] PRIMARY KEY CLUSTERED ([Class_ID]) ON
[Fourth]
GO
```

Used By

[dbo].[Inst_teach_course]
[dbo].[Students]
[InstructorSC].[ShowStudentsInClasses]
[MangerSC].[ShowAllDataFromStudent]
[MangerSC].[ShowInstructorInCourseAndClass]

[dbo].[Courses]

Properties

Property	Value
Collation	Arabic_CI_AS
Row Count (~)	13
Created	11:43:27 PM Sunday, January 14, 2024
Last Modified	12:03:17 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK	C_ID	int	4	NOT NULL	1 - 1
	C_name	nvarchar(50)	100	NOT NULL	
	C_minDegree	int	4	NULL allowed	
	C_maxDegree	int	4	NULL allowed	
	C_Description	nvarchar(150)	300	NULL allowed	

Indexes

Key	Name	Key Columns	Unique
PK	PK_Courses	C_ID	True
	IX_Course_Name	C_name	

Check Constraints

Name	On Column	Constraint
Course_check_maxDegree	C_maxDegree	([C_maxDegree]>=(51) AND [C_maxDegree]<=(100))
Course_check_minDegree	C_minDegree	([C_minDegree]>=(10) AND [C_minDegree]<=(50))

SQL Script

```
CREATE TABLE [dbo].[Courses]
(
[C_ID] [int] NOT NULL IDENTITY(1, 1),
[C_name] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[C_minDegree] [int] NULL,
[C_maxDegree] [int] NULL,
[C_Description] [nvarchar] (150) COLLATE Arabic_CI_AS NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Courses] ADD CONSTRAINT [Course_check_maxDegree] CHECK (([C_maxDegree]>=(51)
AND [C_maxDegree]<=(100)))
GO
ALTER TABLE [dbo].[Courses] ADD CONSTRAINT [Course_check_minDegree] CHECK (([C_minDegree]>=(10)
AND [C_minDegree]<=(50)))
GO
ALTER TABLE [dbo].[Courses] ADD CONSTRAINT [PK_Courses] PRIMARY KEY CLUSTERED ([C_ID]) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Course_Name] ON [dbo].[Courses] ([C_name]) ON [PRIMARY]
GO
```

Used By

[dbo].[Exam]
[dbo].[Inst_teach_course]
[dbo].[Questions]
[InstructorSC].[ShowAllDataFromExam]
[InstructorSC].[ShowDataOFQuestionPool]
[MangerSC].[ShowAllDataFromCourses]
[MangerSC].[ShowInstructorInCourseAndClass]
[MangerSC].[CreateCourses]
[MangerSC].[DeleteCourse]
[MangerSC].[EidtCourse]

[dbo].[Department]

Properties

Property	Value
Collation	Arabic_CI_AS
Row Count (~)	6
Created	11:44:44 PM Sunday, January 14, 2024
Last Modified	1:12:44 AM Monday, January 15, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	dept_ID	int	4	NOT NULL	1 - 1
	dept_name	nvarchar(50)	100	NOT NULL	

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Department	dept_ID	True

SQL Script

```
CREATE TABLE [dbo].[Department]
(
[dept_ID] [int] NOT NULL IDENTITY(1, 1),
[dept_name] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Department] ADD CONSTRAINT [PK_Department] PRIMARY KEY CLUSTERED ([dept_ID])
ON [PRIMARY]
GO
```

Used By

[dbo].[Intake_Depart]
[dbo].[Tracks]
[MangerSC].[ShowDepartmentInIntake]

[dbo].[Exam]

Properties

Property	Value
Collation	Arabic_CI_AS
File Group	Fourth
Row Count (~)	16
Created	10:29:17 PM Monday, January 15, 2024
Last Modified	11:18:52 PM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
	E_ID	int	4	NOT NULL	1 - 1
	E_startTime	time	5	NOT NULL	
	E_endTime	time	5	NOT NULL	
	E_type	nvarchar(50)	100	NOT NULL	
	E_year	int	4	NOT NULL	
	E_allownce	bit	1	NOT NULL	
	Course_ID	int	4	NOT NULL	
	Instructor_ID	int	4	NOT NULL	
	Intake_ID	int	4	NOT NULL	
	Track_ID	int	4	NOT NULL	
	Branch_ID	int	4	NOT NULL	
	E_Date	date	3	NULL allowed	

Indexes

Key	Name	Key Columns	Unique	File Group
	PK_Exam	E_ID	True	Fourth

Check Constraints

Name	On Column	Constraint
Exam_check_Type	E_type	([E_type]='Normal' OR [E_type]='Corrective')
Exam_check_date	E_Date	(CONVERT([date],[E_date],(0))>=getdate())

Exam_check_end		([E_endTime]<>[E_startTime] AND [E_endTime]>[E_startTime])
Exam_check_start		([E_startTime]<>[E_endTime] AND [E_startTime]<[E_endTime])

Foreign Keys

Name	Update	Delete	Columns
FK_Exam_Branchs	Cascade	Cascade	Branch_ID->[dbo].[Branchs].[Branch_ID]
FK_Exam_Courses	Cascade	Cascade	Course_ID->[dbo].[Courses].[C_ID]
FK_Exam_Instructors	Cascade	Cascade	Instructor_ID->[dbo].[Instructors].[Inst_ID]
FK_Exam_Intake	Cascade	Cascade	Intake_ID->[dbo].[Intake].[Intake_ID]
FK_Exam_Tracks	Cascade	Cascade	Track_ID->[dbo].[Tracks].[Track_ID]

SQL Script

```sql
CREATE TABLE [dbo].[Exam]
(
[E_ID] [int] NOT NULL IDENTITY(1, 1),
[E_startTime] [time] NOT NULL,
[E_endTime] [time] NOT NULL,
[E_type] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[E_year] [int] NOT NULL,
[E_allownce] [bit] NOT NULL,
[Course_ID] [int] NOT NULL,
[Instructor_ID] [int] NOT NULL,
[Intake_ID] [int] NOT NULL,
[Track_ID] [int] NOT NULL,
[Branch_ID] [int] NOT NULL,
[E_Date] [date] NULL
) ON [Fourth]
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_check_Type] CHECK (([E_type]='Normal' OR
[E_type]='Corrective'))
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_check_date] CHECK
((CONVERT([date],[E_date],(0))>=getdate()))
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_check_end] CHECK (([E_endTime]<>[E_startTime] AND
[E_endTime]>[E_startTime]))
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_check_start] CHECK (([E_startTime]<>[E_endTime] AND
[E_startTime]<[E_endTime]))
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [PK_Exam] PRIMARY KEY CLUSTERED ([E_ID]) ON [Fourth]
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [FK_Exam_Branchs] FOREIGN KEY ([Branch_ID]) REFERENCES
[dbo].[Branchs] ([Branch_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [FK_Exam_Courses] FOREIGN KEY ([Course_ID]) REFERENCES
[dbo].[Courses] ([C_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
```

```
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [FK_Exam_Instructors] FOREIGN KEY ([Instructor_ID])
REFERENCES [dbo].[Instructors] ([Inst_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [FK_Exam_Intake] FOREIGN KEY ([Intake_ID]) REFERENCES
[dbo].[Intake] ([Intake_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [FK_Exam_Tracks] FOREIGN KEY ([Track_ID]) REFERENCES
[dbo].[Tracks] ([Track_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
```

Uses

[dbo].[Branchs]
[dbo].[Courses]
[dbo].[Instructors]
[dbo].[Intake]
[dbo].[Tracks]

Used By

[dbo].[Answer]
[dbo].[Exam_Question]
[dbo].[Student_Exam]
[InstructorSC].[ShowAllDataFromExam]
[InstructorSC].[ShowQuestionsInExam]
[InstructorSC].[ShowStudentInExam]
[StudentSC].[ShowStudentResults]
[InstructorSC].[CreateExam]
[InstructorSC].[InsertExamQuestions]
[InstructorSC].[InsertExamQuestionsRandomly]
[dbo].[GetAvailableExamsFun]

[dbo].[Exam_Question]

Properties

Property	Value
Row Count (~)	24
Created	12:50:55 AM Monday, January 15, 2024
Last Modified	12:03:17 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
	Exam_ID	int	4	NOT NULL
	Question_ID	int	4	NOT NULL

Indexes

Key	Name	Key Columns	Unique
	PK_Exam_Question	Exam_ID, Question_ID	True

Foreign Keys

Name	Update	Delete	Columns
FK_Exam_Question_Exam	Cascade	Cascade	Exam_ID->[dbo].[Exam].[E_ID]
FK_Exam_Question_Questions			Question_ID->[dbo].[Questions].[Q_ID]

SQL Script

```
CREATE TABLE [dbo].[Exam_Question]
(
[Exam_ID] [int] NOT NULL,
[Question_ID] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam_Question] ADD CONSTRAINT [PK_Exam_Question] PRIMARY KEY CLUSTERED
([Exam_ID], [Question_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam_Question] ADD CONSTRAINT [FK_Exam_Question_Exam] FOREIGN KEY ([Exam_ID])
REFERENCES [dbo].[Exam] ([E_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[Exam_Question] ADD CONSTRAINT [FK_Exam_Question_Questions] FOREIGN KEY
([Question_ID]) REFERENCES [dbo].[Questions] ([Q_ID])
```

```
GO
```

Uses

[dbo].[Exam]
[dbo].[Questions]

Used By

[InstructorSC].[ShowQuestionsInExam]
[InstructorSC].[InsertExamQuestions]
[InstructorSC].[InsertExamQuestionsRandomly]
[StudentSC].[InsertAnswer]
[dbo].[GetAvailableExamsFun]

[dbo].[Inst_teach_course]

Properties

Property	Value
Row Count (~)	6
Created	11:38:34 PM Sunday, January 14, 2024
Last Modified	12:36:16 AM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
	course_ID	int	4	NOT NULL
	Instructor_ID	int	4	NOT NULL
	Class_ID	int	4	NOT NULL
	year	int	4	NOT NULL

Indexes

Key	Name	Key Columns	Unique
	PK_Inst_teach_course	course_ID, Instructor_ID, Class_ID	True

Foreign Keys

Name	Update	Delete	Columns
FK_Inst_teach_couerse_Class	Cascade	Cascade	Class_ID->[dbo].[Class].[Class_ID]
FK_Inst_teach_couerse_Courses	Cascade	Cascade	course_ID->[dbo].[Courses].[C_ID]
FK_Inst_teach_couerse_Instructors	Cascade	Cascade	Instructor_ID->[dbo].[Instructors].[Inst_ID]

SQL Script

```
CREATE TABLE [dbo].[Inst_teach_course]
(
[course_ID] [int] NOT NULL,
[Instructor_ID] [int] NOT NULL,
[Class_ID] [int] NOT NULL,
[year] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Inst_teach_course] ADD CONSTRAINT [PK_Inst_teach_course] PRIMARY KEY CLUSTERED
```

```
([course_ID], [Instructor_ID], [Class_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Inst_teach_course] ADD CONSTRAINT [FK_Inst_teach_couerse_Class] FOREIGN KEY
([Class_ID]) REFERENCES [dbo].[Class] ([Class_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[Inst_teach_course] ADD CONSTRAINT [FK_Inst_teach_couerse_Courses] FOREIGN KEY
([course_ID]) REFERENCES [dbo].[Courses] ([C_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[Inst_teach_course] ADD CONSTRAINT [FK_Inst_teach_couerse_Instructors] FOREIGN
KEY ([Instructor_ID]) REFERENCES [dbo].[Instructors] ([Inst_ID]) ON DELETE CASCADE ON UPDATE
CASCADE
GO
```

Uses

[dbo].[Class]
[dbo].[Courses]
[dbo].[Instructors]

Used By

[MangerSC].[ShowInstructorInCourseAndClass]
[InstructorSC].[CreateExam]
[MangerSC].[CreateInstructorINCourse]
[MangerSC].[EidtInstForEachCourse]

[dbo].[Instructors]

Properties

Property	Value
Collation	Arabic_CI_AS
File Group	Third
Row Count (~)	5
Created	1:11:54 AM Monday, January 15, 2024
Last Modified	11:58:53 PM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
🔑 PK C	Inst_ID	int	4	NOT NULL	1 - 1
🖧	Inst_name	nvarchar(50)	100	NOT NULL	
🖧	Inst_email	nvarchar(50)	100	NOT NULL	
	Inst_password	nvarchar(50)	100	NOT NULL	
🔑 FK	manager_id	int	4	NULL allowed	

Indexes

Key	Name	Key Columns	Unique	File Group
🔑 PK C	PK_Instructors	Inst_ID	True	Third
	IX_Uniqe_Instructor_Email	Inst_email	True	
	IX_Instructor_Name	Inst_name		Third

Foreign Keys

Name	Columns
FK_Instructors_Instructors	manager_id->[dbo].[Instructors].[Inst_ID]

SQL Script

```
CREATE TABLE [dbo].[Instructors]
(
[Inst_ID] [int] NOT NULL IDENTITY(1, 1),
[Inst_name] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[Inst_email] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[Inst_password] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[manager_id] [int] NULL
) ON [Third]
GO
ALTER TABLE [dbo].[Instructors] ADD CONSTRAINT [PK_Instructors] PRIMARY KEY CLUSTERED ([Inst_ID])
ON [Third]
GO
ALTER TABLE [dbo].[Instructors] ADD CONSTRAINT [IX_Uniqe_Instructor_Email] UNIQUE NONCLUSTERED
([Inst_email]) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Instructor_Name] ON [dbo].[Instructors] ([Inst_name]) ON [Third]
GO
ALTER TABLE [dbo].[Instructors] ADD CONSTRAINT [FK_Instructors_Instructors] FOREIGN KEY
([manager_id]) REFERENCES [dbo].[Instructors] ([Inst_ID])
GO
```

Used By

[dbo].[Exam]
[dbo].[Inst_teach_course]
[InstructorSC].[ShowAllDataFromExam]
[MangerSC].[ShowAllDataFromInstaructors]
[MangerSC].[ShowInstructorInCourseAndClass]
[MangerSC].[CreateInstructor]
[MangerSC].[DeleteInstructor]
[MangerSC].[EidtInstructor]

[dbo].[Intake]

Properties

Property	Value
Collation	Arabic_CI_AS
File Group	Third
Row Count (~)	7
Created	1:12:25 AM Monday, January 15, 2024
Last Modified	12:38:43 AM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Intake_ID	int	4	NOT NULL	1 - 1
	Intake_name	nvarchar(50)	100	NOT NULL	

Indexes

Key	Name	Key Columns	Unique	File Group
PK C	PK_Intake	Intake_ID	True	Third

SQL Script

```
CREATE TABLE [dbo].[Intake]
(
[Intake_ID] [int] NOT NULL IDENTITY(1, 1),
[Intake_name] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL
) ON [Third]
GO
ALTER TABLE [dbo].[Intake] ADD CONSTRAINT [PK_Intake] PRIMARY KEY CLUSTERED ([Intake_ID]) ON
[Third]
GO
```

Used By

[dbo].[Exam]
[dbo].[Intake_Branch]
[dbo].[Intake_Depart]
[dbo].[Students]
[InstructorSC].[ShowAllDataFromExam]

[MangerSC].[ShowAllDataFromStudent]
[MangerSC].[ShowDepartmentInIntake]

[dbo].[Intake_Branch]

Properties

Property	Value
Row Count (~)	13
Created	12:14:16 AM Monday, January 15, 2024
Last Modified	1:12:26 AM Monday, January 15, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK	bran_ID	int	4	NOT NULL
PK FK	intak_ID	int	4	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK	PK_Intake_Branch	bran_ID, intak_ID	True

Foreign Keys

Name	Columns
FK_Intake_Branch_Branchs	bran_ID->[dbo].[Branchs].[Branch_ID]
FK_Intake_Branch_Intake	intak_ID->[dbo].[Intake].[Intake_ID]

SQL Script

```
CREATE TABLE [dbo].[Intake_Branch]
(
[bran_ID] [int] NOT NULL,
[intak_ID] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Intake_Branch] ADD CONSTRAINT [PK_Intake_Branch] PRIMARY KEY CLUSTERED
([bran_ID], [intak_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Intake_Branch] ADD CONSTRAINT [FK_Intake_Branch_Branchs] FOREIGN KEY ([bran_-
ID]) REFERENCES [dbo].[Branchs] ([Branch_ID])
GO
ALTER TABLE [dbo].[Intake_Branch] ADD CONSTRAINT [FK_Intake_Branch_Intake] FOREIGN KEY ([intak_-
ID]) REFERENCES [dbo].[Intake] ([Intake_ID])
```

```
GO
```

Uses

[dbo].[Branchs]
[dbo].[Intake]

[dbo].[Intake_Depart]

Properties

Property	Value
Row Count (~)	7
Created	12:02:49 AM Monday, January 15, 2024
Last Modified	1:12:26 AM Monday, January 15, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
	Depart_ID	int	4	NOT NULL
	Intake_id	int	4	NOT NULL

Indexes

Key	Name	Key Columns	Unique
	PK_Intake_Depart	Depart_ID, Intake_id	True

Foreign Keys

Name	Columns
FK_Intake_Depart_Department	Depart_ID->[dbo].[Department].[dept_ID]
FK_Intake_Depart_Intake	Intake_id->[dbo].[Intake].[Intake_ID]

SQL Script

```
CREATE TABLE [dbo].[Intake_Depart]
(
[Depart_ID] [int] NOT NULL,
[Intake_id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Intake_Depart] ADD CONSTRAINT [PK_Intake_Depart] PRIMARY KEY CLUSTERED
([Depart_ID], [Intake_id]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Intake_Depart] ADD CONSTRAINT [FK_Intake_Depart_Department] FOREIGN KEY
([Depart_ID]) REFERENCES [dbo].[Department] ([dept_ID])
GO
ALTER TABLE [dbo].[Intake_Depart] ADD CONSTRAINT [FK_Intake_Depart_Intake] FOREIGN KEY
([Intake_id]) REFERENCES [dbo].[Intake] ([Intake_ID])
```

```
GO
```

Uses

[dbo].[Department]
[dbo].[Intake]

Used By

[MangerSC].[ShowDepartmentInIntake]
[MangerSC].[EidtIntake_Depart]

[dbo].[Questions]

Properties

Property	Value
Collation	Arabic_CI_AS
File Group	Third
Row Count (~)	26
Created	12:03:16 AM Wednesday, January 17, 2024
Last Modified	12:03:17 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity	Default
	Q_ID	int	4	NOT NULL	1 - 1	
	Q_Text	nvarchar(250)	500	NOT NULL		
	Q_correctAns	nvarchar(200)	400	NOT NULL		
	Q_type	nvarchar(50)	100	NOT NULL		
	Course_ID	int	4	NOT NULL		
	Q_Mark	int	4	NOT NULL		((1))

Indexes

Key	Name	Key Columns	Unique	File Group
	PK_Questions	Q_ID	True	Third

Check Constraints

Name	On Column	Constraint
Question_check_Type	Q_type	([Q_type]='MCQ' OR [Q_type]='T/F' OR [Q_type]='Written')
Questions_Defult_Mark	Q_Mark	([Q_Mark]>=(1))

Foreign Keys

Name	Columns
FK_Questions_Courses	Course_ID->[dbo].[Courses].[C_ID]

SQL Script

```
CREATE TABLE [dbo].[Questions]
(
[Q_ID] [int] NOT NULL IDENTITY(1, 1),
[Q_Text] [nvarchar] (250) COLLATE Arabic_CI_AS NOT NULL,
[Q_correctAns] [nvarchar] (200) COLLATE Arabic_CI_AS NOT NULL,
[Q_type] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[Course_ID] [int] NOT NULL,
[Q_Mark] [int] NOT NULL CONSTRAINT [DF_Questions_Q_Mark] DEFAULT ((1))
) ON [Third]
GO
ALTER TABLE [dbo].[Questions] ADD CONSTRAINT [Question_check_Type] CHECK (([Q_type]='MCQ' OR
[Q_type]='T/F' OR [Q_type]='Written'))
GO
ALTER TABLE [dbo].[Questions] ADD CONSTRAINT [Questions_Defult_Mark] CHECK (([Q_Mark]>=(1)))
GO
ALTER TABLE [dbo].[Questions] ADD CONSTRAINT [PK_Questions] PRIMARY KEY CLUSTERED ([Q_ID]) ON
[Third]
GO
ALTER TABLE [dbo].[Questions] ADD CONSTRAINT [FK_Questions_Courses] FOREIGN KEY ([Course_ID])
REFERENCES [dbo].[Courses] ([C_ID])
GO
```

Uses

[dbo].[Courses]

Used By

[dbo].[Answer]
[dbo].[Exam_Question]
[InstructorSC].[ShowDataOFQuestionPool]
[InstructorSC].[ShowQuestionsInExam]
[InstructorSC].[CreateQuestion]
[InstructorSC].[DeleteQuestion]
[InstructorSC].[EditQuestions]
[InstructorSC].[EidtQuestionsMark]
[InstructorSC].[InsertExamQuestions]
[InstructorSC].[InsertExamQuestionsRandomly]
[dbo].[CalculateTotalCorrectAnswers]
[dbo].[GetAvailableExamsFun]

[dbo].[Student_Exam]

Properties

Property	Value
Collation	Arabic_CI_AS
Row Count (~)	10
Created	12:53:08 AM Monday, January 15, 2024
Last Modified	12:38:43 AM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK	Student_ID	int	4	NOT NULL
PK FK	Exam_ID	int	4	NOT NULL
	Results	nvarchar(50)	100	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK	PK_Student_Exam	Student_ID, Exam_ID	True

Foreign Keys

Name	Update	Delete	Columns
FK_Student_Exam_Exam	Cascade	Cascade	Exam_ID->[dbo].[Exam].[E_ID]
FK_Student_Exam_Students			Student_ID->[dbo].[Students].[Std_ID]

SQL Script

```
CREATE TABLE [dbo].[Student_Exam]
(
[Student_ID] [int] NOT NULL,
[Exam_ID] [int] NOT NULL,
[Results] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Exam] ADD CONSTRAINT [PK_Student_Exam] PRIMARY KEY CLUSTERED
([Student_ID], [Exam_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Exam] ADD CONSTRAINT [FK_Student_Exam_Exam] FOREIGN KEY ([Exam_ID])
```

```
REFERENCES [dbo].[Exam] ([E_ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[Student_Exam] ADD CONSTRAINT [FK_Student_Exam_Students] FOREIGN KEY
([Student_ID]) REFERENCES [dbo].[Students] ([Std_ID])
GO
```

Uses

[dbo].[Exam]
[dbo].[Students]

Used By

[InstructorSC].[ShowStudentInExam]
[StudentSC].[ShowStudentResults]
[dbo].[UpdateStudentExamResults]
[InstructorSC].[insertStudentToExams]
[StudentSC].[InsertAnswer]

[dbo].[Students]

Properties

Property	Value
Collation	Arabic_CI_AS
File Group	Secoundary
Row Count (~)	20
Created	1:10:49 AM Monday, January 15, 2024
Last Modified	8:45:37 PM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
	Std_ID	int	4	NOT NULL	1 - 1
	Std_name	nvarchar(50)	100	NOT NULL	
	Std_email	nvarchar(50)	100	NOT NULL	
	Std_password	nvarchar(50)	100	NOT NULL	
	Intake_id	int	4	NOT NULL	
	barch_id	int	4	NOT NULL	
	track_id	int	4	NOT NULL	
	class_id	int	4	NOT NULL	

Indexes

Key	Name	Key Columns	Unique	File Group
	PK_Students'	Std_ID	True	Secoundary
	IX_Students'	Std_email	True	
	IX_Students	Std_name		

Foreign Keys

Name	Columns
FK_Students'_Branchs	barch_id->[dbo].[Branchs].[Branch_ID]
FK_Students'_Class	class_id->[dbo].[Class].[Class_ID]
FK_Students'_Intake	Intake_id->[dbo].[Intake].[Intake_ID]
FK_Students'_Tracks	track_id->[dbo].[Tracks].[Track_ID]

SQL Script

```
CREATE TABLE [dbo].[Students]
(
[Std_ID] [int] NOT NULL IDENTITY(1, 1),
[Std_name] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[Std_email] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[Std_password] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[Intake_id] [int] NOT NULL,
[barch_id] [int] NOT NULL,
[track_id] [int] NOT NULL,
[class_id] [int] NOT NULL
) ON [Secoundary]
GO
ALTER TABLE [dbo].[Students] ADD CONSTRAINT [PK_Students'] PRIMARY KEY CLUSTERED ([Std_ID]) ON
[Secoundary]
GO
ALTER TABLE [dbo].[Students] ADD CONSTRAINT [IX_Students'] UNIQUE NONCLUSTERED ([Std_email]) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Students] ON [dbo].[Students] ([Std_name]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Students] ADD CONSTRAINT [FK_Students'_Branchs] FOREIGN KEY ([barch_id])
REFERENCES [dbo].[Branchs] ([Branch_ID])
GO
ALTER TABLE [dbo].[Students] ADD CONSTRAINT [FK_Students'_Class] FOREIGN KEY ([class_id])
REFERENCES [dbo].[Class] ([Class_ID])
GO
ALTER TABLE [dbo].[Students] ADD CONSTRAINT [FK_Students'_Intake] FOREIGN KEY ([Intake_id])
REFERENCES [dbo].[Intake] ([Intake_ID])
GO
ALTER TABLE [dbo].[Students] ADD CONSTRAINT [FK_Students'_Tracks] FOREIGN KEY ([track_id])
REFERENCES [dbo].[Tracks] ([Track_ID])
GO
```

Uses

[dbo].[Branchs]
[dbo].[Class]
[dbo].[Intake]
[dbo].[Tracks]

Used By

[dbo].[Answer]
[dbo].[Student_Exam]
[InstructorSC].[ShowStudentInExam]
[InstructorSC].[ShowStudentsInClasses]
[MangerSC].[ShowAllDataFromStudent]
[StudentSC].[ShowStudentResults]
[InstructorSC].[GetStudentByID]

[InstructorSC].[insertStudentToExams]
[MangerSC].[CreateStudent]
[MangerSC].[DeleteStudent]
[MangerSC].[EditStudent]

[dbo].[Tracks]

Properties

Property	Value
Collation	Arabic_CI_AS
File Group	Third
Row Count (~)	8
Created	1:12:44 AM Monday, January 15, 2024
Last Modified	12:38:43 AM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Track_ID	int	4	NOT NULL	1 - 1
	Track_name	nvarchar(50)	100	NOT NULL	
FK	Department_ID	int	4	NULL allowed	

Indexes

Key	Name	Key Columns	Unique	File Group
PK C	PK_Tracks	Track_ID	True	Third

Foreign Keys

Name	Columns
FK_Tracks_Department	Department_ID->[dbo].[Department].[dept_ID]

SQL Script

```
CREATE TABLE [dbo].[Tracks]
(
[Track_ID] [int] NOT NULL IDENTITY(1, 1),
[Track_name] [nvarchar] (50) COLLATE Arabic_CI_AS NOT NULL,
[Department_ID] [int] NULL
) ON [Third]
GO
ALTER TABLE [dbo].[Tracks] ADD CONSTRAINT [PK_Tracks] PRIMARY KEY CLUSTERED ([Track_ID]) ON
[Third]
GO
ALTER TABLE [dbo].[Tracks] ADD CONSTRAINT [FK_Tracks_Department] FOREIGN KEY ([Department_ID])
```

```
REFERENCES [dbo].[Department] ([dept_ID])
GO
```

Uses

[dbo].[Department]

Used By

[dbo].[Exam]
[dbo].[Students]
[InstructorSC].[ShowAllDataFromExam]
[MangerSC].[ShowAllDataFromStudent]

Views

Objects

Name
InstructorSC.ShowAllDataFromExam
InstructorSC.ShowDataOFQuestionPool
InstructorSC.ShowQuestionsInExam
InstructorSC.ShowStudentInExam
InstructorSC.ShowStudentsInClasses
MangerSC.ShowAllDataFromBranch
MangerSC.ShowAllDataFromCourses
MangerSC.ShowAllDataFromInstaructors
MangerSC.ShowAllDataFromStudent
MangerSC.ShowDepartmentInIntake
MangerSC.ShowInstructorInCourseAndClass
StudentSC.ShowStudentResults

[InstructorSC].[ShowAllDataFromExam]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	9:14:00 PM Tuesday, January 16, 2024
Last Modified	9:30:21 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
E_ID	int	4
E_startTime	time	5
E_endTime	time	5
E_type	nvarchar(50)	100
E_year	int	4
E_allownce	bit	1
C_name	nvarchar(50)	100
Inst_name	nvarchar(50)	100
Intake_name	nvarchar(50)	100
Track_name	nvarchar(50)	100
Branch_name	nvarchar(50)	100
E_Date	date	3

SQL Script

```
                  create view [InstructorSC].[ShowAllDataFromExam]
                  as

SELECT dbo.Exam.E_ID, dbo.Exam.E_startTime, dbo.Exam.E_endTime, dbo.Exam.E_type, dbo.Exam.E_year,
dbo.Exam.E_allownce, dbo.Courses.C_name, dbo.Instructors.Inst_name, dbo.Intake.Intake_name,
dbo.Tracks.Track_name,
                  dbo.Branchs.Branch_name, dbo.Exam.E_Date
FROM     dbo.Branchs INNER JOIN
                  dbo.Exam ON dbo.Branchs.Branch_ID = dbo.Exam.Branch_ID INNER JOIN
                  dbo.Courses ON dbo.Exam.Course_ID = dbo.Courses.C_ID INNER JOIN
                  dbo.Instructors ON dbo.Exam.Instructor_ID = dbo.Instructors.Inst_ID INNER JOIN
                  dbo.Intake ON dbo.Exam.Intake_ID = dbo.Intake.Intake_ID INNER JOIN
                  dbo.Tracks ON dbo.Exam.Track_ID = dbo.Tracks.Track_ID
```

```
GO
```

Uses

[dbo].[Branchs]
[dbo].[Courses]
[dbo].[Exam]
[dbo].[Instructors]
[dbo].[Intake]
[dbo].[Tracks]
InstructorSC

[InstructorSC].[ShowDataOFQuestionPool]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	2:13:54 PM Tuesday, January 16, 2024
Last Modified	9:30:21 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Q_Text	nvarchar(250)	500
Q_correctAns	nvarchar(200)	400
Q_type	nvarchar(50)	100
C_name	nvarchar(50)	100
C_minDegree	int	4
C_maxDigree	int	4
Q_Mark	int	4

SQL Script

```
create view [InstructorSC].[ShowDataOFQuestionPool]
AS
SELECT TOP (200) Questions.Q_Text, Questions.Q_correctAns, Questions.Q_type, Courses.C_name,
Courses.C_minDegree, Courses.C_maxDigree, Questions.Q_Mark
FROM     Questions INNER JOIN
                  Courses ON Questions.Course_ID = Courses.C_ID
GO
```

Uses

[dbo].[Courses]
[dbo].[Questions]
InstructorSC

[InstructorSC].[ShowQuestionsInExam]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	9:18:47 PM Tuesday, January 16, 2024
Last Modified	9:30:21 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
E_ID	int	4
Q_Text	nvarchar(250)	500
Q_correctAns	nvarchar(200)	400
Q_type	nvarchar(50)	100
Q_Mark	int	4

SQL Script

```
CREATE view [InstructorSC].[ShowQuestionsInExam]
as
SELECT  dbo.Exam.E_ID, dbo.Questions.Q_Text, dbo.Questions.Q_correctAns, dbo.Questions.Q_type,
dbo.Questions.Q_Mark
FROM     dbo.Exam INNER JOIN
                  dbo.Exam_Question ON dbo.Exam.E_ID = dbo.Exam_Question.Exam_ID INNER JOIN
                  dbo.Questions ON dbo.Exam_Question.Question_ID = dbo.Questions.Q_ID
GO
```

Uses

[dbo].[Exam]
[dbo].[Exam_Question]
[dbo].[Questions]
InstructorSC

[InstructorSC].[ShowStudentInExam]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	9:23:05 PM Tuesday, January 16, 2024
Last Modified	9:30:21 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Std_name	nvarchar(50)	100
E_ID	int	4
Results	nvarchar(50)	100

SQL Script

```
  Create view [InstructorSC].[ShowStudentInExam]
                as
                SELECT dbo.Students.Std_name, dbo.Exam.E_ID, dbo.Student_Exam.Results
FROM     dbo.Exam INNER JOIN
                dbo.Student_Exam ON dbo.Exam.E_ID = dbo.Student_Exam.Exam_ID INNER JOIN
                dbo.Students ON dbo.Student_Exam.Student_ID = dbo.Students.Std_ID
GO
```

Uses

[dbo].[Exam]
[dbo].[Student_Exam]
[dbo].[Students]
InstructorSC

[InstructorSC].[ShowStudentsInClasses]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	9:16:14 PM Tuesday, January 16, 2024
Last Modified	9:30:21 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Std_name	nvarchar(50)	100
Class_name	nvarchar(50)	100

SQL Script

```
create view [InstructorSC].[ShowStudentsInClasses]
as
SELECT dbo.Students.Std_name, dbo.Class.Class_name
FROM   dbo.Class INNER JOIN dbo.Students
ON dbo.Class.Class_ID = dbo.Students.class_id
GO
```

Uses

[dbo].[Class]
[dbo].[Students]
InstructorSC

[MangerSC].[ShowAllDataFromBranch]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	9:05:38 PM Tuesday, January 16, 2024
Last Modified	9:29:01 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)	Identity
Branch_ID	int	4	0 - 0
Branch_name	nvarchar(50)	100	
Branch_address	nvarchar(50)	100	

SQL Script

```
                create view [MangerSC].[ShowAllDataFromBranch]
                as
                select * from Branchs
GO
```

Uses

[dbo].[Branchs]
MangerSC

[MangerSC].[ShowAllDataFromCourses]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	8:54:33 PM Tuesday, January 16, 2024
Last Modified	9:29:01 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)	Identity
C_ID	int	4	0 - 0
C_name	nvarchar(50)	100	
C_minDegree	int	4	
C_maxDigree	int	4	
C_Description	nvarchar(150)	300	

SQL Script

```
create view [MangerSC].[ShowAllDataFromCourses]
as
select * from Courses
GO
```

Uses

[dbo].[Courses]
MangerSC

[MangerSC].[ShowAllDataFromInstaructors]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	8:52:01 PM Tuesday, January 16, 2024
Last Modified	9:29:01 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)	Identity
Inst_name	nvarchar(50)	100	
Inst_ID	int	4	0 - 0
Inst_email	nvarchar(50)	100	
Inst_password	nvarchar(50)	100	
MangerName	nvarchar(50)	100	

SQL Script

```
create view [MangerSC].[ShowAllDataFromInstaructors]
as
SELECT i.Inst_name, i.Inst_ID, i.Inst_email, i.Inst_password,
(select m.Inst_name from Instructors m where i.manager_id = m.Inst_ID ) as MangerName
FROM     dbo.Instructors i
GO
```

Uses

[dbo].[Instructors]
MangerSC

[MangerSC].[ShowAllDataFromStudent]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	6:08:35 AM Monday, January 15, 2024
Last Modified	9:29:01 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Std_name	nvarchar(50)	100
Std_email	nvarchar(50)	100
Std_password	nvarchar(50)	100
Intake_name	nvarchar(50)	100
Branch_name	nvarchar(50)	100
Track_name	nvarchar(50)	100
Class_name	nvarchar(50)	100

SQL Script

```
create view [MangerSC].[ShowAllDataFromStudent]
as
SELECT dbo.Students.Std_name, dbo.Students.Std_email, dbo.Students.Std_password,
dbo.Intake.Intake_name, dbo.Branchs.Branch_name, dbo.Tracks.Track_name, dbo.Class.Class_name
FROM     dbo.Branchs INNER JOIN
                  dbo.Students ON dbo.Branchs.Branch_ID = dbo.Students.barch_id INNER JOIN
                  dbo.Class ON dbo.Students.class_id = dbo.Class.Class_ID INNER JOIN
                  dbo.Intake ON dbo.Students.Intake_id = dbo.Intake.Intake_ID INNER JOIN
                  dbo.Tracks ON dbo.Students.track_id = dbo.Tracks.Track_ID
GO
```

Uses

[dbo].[Branchs]
[dbo].[Class]
[dbo].[Intake]

[dbo].[Students]

[dbo].[Tracks]

MangerSC

[MangerSC].[ShowDepartmentInIntake]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	9:08:17 PM Tuesday, January 16, 2024
Last Modified	9:29:01 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Intake_name	nvarchar(50)	100
dept_name	nvarchar(50)	100

SQL Script

```
      create view [MangerSC].[ShowDepartmentInIntake]
                  as
                  SELECT  dbo.Intake.Intake_name, dbo.Department.dept_name
                  FROM     dbo.Department INNER JOIN
                  dbo.Intake_Depart ON dbo.Department.dept_ID = dbo.Intake_Depart.Depart_ID INNER
JOIN
                  dbo.Intake ON dbo.Intake_Depart.Intake_id = dbo.Intake.Intake_ID
GO
```

Uses

[dbo].[Department]
[dbo].[Intake]
[dbo].[Intake_Depart]
MangerSC

[MangerSC].[ShowInstructorInCourseAndClass]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	8:58:50 PM Tuesday, January 16, 2024
Last Modified	9:29:01 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Inst_name	nvarchar(50)	100
Course_Name	nvarchar(50)	100
Class_name	nvarchar(50)	100
year	int	4

SQL Script

```
create view [MangerSC].[ShowInstructorInCourseAndClass]
as
SELECT dbo.Instructors.Inst_name, dbo.Courses.C_name as Course_Name, dbo.Class.Class_name,
dbo.Inst_teach_course.year
FROM     dbo.Class INNER JOIN
                  dbo.Inst_teach_course ON dbo.Class.Class_ID = dbo.Inst_teach_course.Class_ID
INNER JOIN
                  dbo.Courses ON dbo.Inst_teach_course.course_ID = dbo.Courses.C_ID INNER JOIN
                  dbo.Instructors ON dbo.Inst_teach_course.Instructor_ID = dbo.Instructors.Inst_-
ID

GO
```

Uses

[dbo].[Class]
[dbo].[Courses]
[dbo].[Inst_teach_course]
[dbo].[Instructors]
MangerSC

[StudentSC].[ShowStudentResults]

Properties

Property	Value
Collation	Arabic_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	9:27:02 PM Tuesday, January 16, 2024
Last Modified	9:30:41 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Std_name	nvarchar(50)	100
E_ID	int	4
Results	nvarchar(50)	100

SQL Script

```
          create view [StudentSC].[ShowStudentResults]
                  as
                  SELECT dbo.Students.Std_name, dbo.Exam.E_ID, dbo.Student_Exam.Results
                  FROM dbo.Exam INNER JOIN
                  dbo.Student_Exam ON dbo.Exam.E_ID = dbo.Student_Exam.Exam_ID INNER JOIN
                  dbo.Students ON dbo.Student_Exam.Student_ID = dbo.Students.Std_ID
GO
```

Uses

[dbo].[Exam]
[dbo].[Student_Exam]
[dbo].[Students]
StudentSC

Stored Procedures

Objects

Name
dbo.UpdateStudentExamResults
InstructorSC.CreateExam
InstructorSC.CreateQuestion
InstructorSC.DeleteQuestion
InstructorSC.EditQuestions
InstructorSC.EidtQuestionsMark
InstructorSC.GetStudentByID
InstructorSC.InsertExamQuestions
InstructorSC.InsertExamQuestionsRandomly
InstructorSC.insertStudentToExams
MangerSC.CreateBranch
MangerSC.CreateCourses
MangerSC.CreateInstructor
MangerSC.CreateInstructorINCourse
MangerSC.CreateStudent
MangerSC.DeleteCourse
MangerSC.DeleteInstructor
MangerSC.DeleteStudent
MangerSC.EditStudent
MangerSC.EidtBranch
MangerSC.EidtCourse
MangerSC.EidtInstForEachCourse
MangerSC.EidtInstructor
MangerSC.EidtIntake_Depart
StudentSC.GetAvailableExams
StudentSC.InsertAnswer

[dbo].[UpdateStudentExamResults]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StudentID	int	4
@ExamID	int	4

SQL Script

```
CREATE PROCEDURE [dbo].[UpdateStudentExamResults]
    @StudentID INT,
    @ExamID INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @TotalCorrectAnswers INT;

    -- Calculate total correct answers using the function
    SET @TotalCorrectAnswers = dbo.CalculateTotalCorrectAnswers(@ExamID, @StudentID);

    -- Update the results in Student_Exam table
    UPDATE dbo.Student_Exam
    SET Results = CONVERT(NVARCHAR(50), @TotalCorrectAnswers)
    WHERE Student_ID = @StudentID
    AND Exam_ID = @ExamID;
END;
GO
```

Uses

[dbo].[Student_Exam]
[dbo].[CalculateTotalCorrectAnswers]

[InstructorSC].[CreateExam]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartTime	datetime	8
@EndTime	datetime	8
@Type	nvarchar(50)	100
@Year	int	4
@Allowance	bit	1
@CourseID	int	4
@InstructorID	int	4
@IntakeID	int	4
@TrackID	int	4
@BranchID	int	4
@Date	date	3

SQL Script

```
CREATE   PROCEDURE [InstructorSC].[CreateExam]
    @StartTime DATETIME,
    @EndTime DATETIME,
    @Type NVARCHAR(50),
    @Year INT,
    @Allowance BIT,
    @CourseID INT,
    @InstructorID INT,
    @IntakeID INT,
    @TrackID INT,
    @BranchID INT,
    @Date date
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
```

```
        IF EXISTS (
            SELECT 1
            FROM dbo.Inst_teach_course
            WHERE course_ID = @CourseID
            AND Instructor_ID = @InstructorID
        )
        BEGIN
            -- Perform the insertion
            INSERT INTO [dbo].[Exam]
                ([E_startTime], [E_endTime], [E_type], [E_year],[E_allownce] , [Course_ID],
                 [Instructor_ID], [Intake_ID], [Track_ID], [Branch_ID] ,[E_Date] )
            VALUES
                (@StartTime, @EndTime, @Type, @Year, @Allowance, @CourseID, @InstructorID,
                 @IntakeID, @TrackID, @BranchID , @date);

            PRINT 'Exam Created successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Instructor is not assigned to the specified course. Exam creation failed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while creating the Exam. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Exam]
[dbo].[Inst_teach_course]
InstructorSC

[InstructorSC].[CreateQuestion]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@text	nvarchar(200)	400
@correctAns	nvarchar(200)	400
@type	nvarchar(50)	100
@IdCourse	int	4

SQL Script

```
CREATE PROC [InstructorSC].[CreateQuestion]
    @text nvarchar(200),
    @correctAns nvarchar(200),
    @type nvarchar(50),
    @IdCourse int

AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Perform the insertion
        INSERT INTO [dbo].[Questions]
               ([Q_Text],[Q_correctAns],[Q_type],[Course_ID])
         VALUES
               (@text,@correctAns,@type,@IdCourse );

        PRINT 'Question added successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while Created the Question. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Questions]

InstructorSC

[InstructorSC].[DeleteQuestion]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@id	int	4

SQL Script

```
CREATE PROC [InstructorSC].[DeleteQuestion]
    @id INT
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before deleting
        IF EXISTS (SELECT 1 FROM [dbo].[Questions] WHERE Q_ID = @id)
        BEGIN
            -- Perform the deletion
            DELETE FROM [dbo].[Questions]
            WHERE Q_ID = @id;

            PRINT 'Question deleted successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Question not found. No deletion performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while deleting the Question. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Questions]

InstructorSC

[InstructorSC].[EditQuestions]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@correct	nvarchar(200)	400
@id	int	4

SQL Script

```
CREATE PROC [InstructorSC].[EditQuestions]
@correct nvarchar(200),
@id int
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before updating
        IF EXISTS (SELECT 1 FROM [dbo].[Questions] WHERE Q_ID = @id)
        BEGIN
            -- Perform the update
            UPDATE [dbo].[Questions]
             set  Q_correctAns = @correct
            where Q_ID = @id

            PRINT 'Correct Answer Question information updated successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Correct Answer Question not found. No update performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while updating the Correct Answer Question information. Error: '
+ ERROR_MESSAGE();
    END CATCH
```

```
END;
GO
```

Uses

[dbo].[Questions]
InstructorSC

[InstructorSC].[EidtQuestionsMark]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Mark	int	4
@id	int	4

SQL Script

```
CREATE   PROC [InstructorSC].[EidtQuestionsMark]
@Mark int,
@id int
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before updating
        IF EXISTS (SELECT 1 FROM [dbo].[Questions] WHERE Q_ID = @id)
        BEGIN
            -- Perform the update
            UPDATE [dbo].[Questions]
             set  Q_Mark = @Mark
            where Q_ID = @id

            PRINT 'Question Mark updated successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Question not found. No update performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while updating the Question Mark. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Questions]

InstructorSC

[InstructorSC].[GetStudentByID]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@id	int	4

SQL Script

```sql
create Proc [InstructorSC].[GetStudentByID]
    @id INT
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Retrieve the student by ID
        SELECT *
        FROM Students
        WHERE Std_ID = @id;

        -- Check if any rows were affected
        IF @@ROWCOUNT = 0
        BEGIN
            PRINT 'No student found with the specified ID.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while retrieving the student. Error: ' + ERROR_MESSAGE();
    END CATCH
END;

GO
```

Uses

[dbo].[Students]

InstructorSC

[InstructorSC].[InsertExamQuestions]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ExamID	int	4
@NumberOfQuestions	int	4
@QuestionIDs	varchar(max)	max

SQL Script

```
CREATE PROCEDURE [InstructorSC].[InsertExamQuestions]
    @ExamID INT,
    @NumberOfQuestions INT,
    @QuestionIDs VARCHAR(MAX)
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @QuestionIDList TABLE (ID INT);

    -- Split the comma-separated string of Question IDs and insert into a table variable
    INSERT INTO @QuestionIDList (ID)
    SELECT value
    FROM STRING_SPLIT(@QuestionIDs, ',');

    -- Check if the number of questions provided matches the actual count
    IF @NumberOfQuestions <> (SELECT COUNT(*) FROM @QuestionIDList)
    BEGIN
        PRINT 'Number of questions does not match the provided count.';
        RETURN;
    END

    -- Check if the provided Question IDs exist in the Questions table
    IF EXISTS (
        SELECT 1
        FROM @QuestionIDList ql
        WHERE NOT EXISTS (
            SELECT 1
            FROM dbo.Questions q
            WHERE q.Q_ID = ql.ID
        )
    )
    BEGIN
        PRINT 'One or more Question IDs do not exist in the Questions table.';
        RETURN;
    END

    -- Check if the Course_ID of selected questions matches the Course_ID of the specified exam
    IF EXISTS (
        SELECT 1
        FROM @QuestionIDList ql
        INNER JOIN dbo.Questions q ON ql.ID = q.Q_ID
        WHERE q.Course_ID <> (SELECT e.Course_ID FROM dbo.Exam e WHERE e.E_ID = @ExamID)
    )
    BEGIN
        PRINT 'Course_ID of selected questions does not match the Course_ID of the specified
exam.';
        RETURN;
    END

    -- Insert data into Exam_Question table
    INSERT INTO dbo.Exam_Question (Exam_ID, Question_ID)
    SELECT @ExamID, ID
    FROM @QuestionIDList;

    PRINT 'Data inserted into Exam_Question successfully.';
END;
GO
```

Uses

[dbo].[Exam]
[dbo].[Exam_Question]
[dbo].[Questions]
InstructorSC

[InstructorSC].[InsertExamQuestionsRandomly]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ExamID	int	4
@NumberOfQuestions	int	4

SQL Script

```
CREATE     PROCEDURE [InstructorSC].[InsertExamQuestionsRandomly]
    @ExamID INT,
    @NumberOfQuestions INT
    --@CourseId int
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @QuestionIDList TABLE (Q_id INt);

INSERT INTO @QuestionIDList (Q_id)
SELECT
    Questions.Q_ID
FROM
    Questions

WHERE
    Questions.Course_ID = (select Exam.Course_ID from Exam where Exam.E_ID =@ExamID)
ORDER BY
    NEWID();

    -- Check if the number of questions provided matches the actual count
    IF @NumberOfQuestions > (SELECT COUNT(*) FROM @QuestionIDList)
    BEGIN
        PRINT 'Number of questions does not supported in this course please add question or
minimize question number';
        RETURN;
    END
    -- Insert data into Exam_Question table
    INSERT INTO dbo.Exam_Question (Exam_ID, Question_ID)
    SELECT @ExamID,Q_id
    FROM @QuestionIDList;

    PRINT 'Data inserted into Exam_Question successfully.';
END;
GO
```

Uses

[dbo].[Exam]
[dbo].[Exam_Question]
[dbo].[Questions]
InstructorSC

[InstructorSC].[insertStudentToExams]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ExamID	int	4
@NumberOfStudents	int	4
@StudentIDs	varchar(max)	max

SQL Script

```
CREATE PROCEDURE [InstructorSC].[insertStudentToExams]
    @ExamID INT,
    @NumberOfStudents INT,
    @StudentIDs VARCHAR(MAX)
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StudentIDsList TABLE (ID INT);

    -- Split the comma-separated string of Question IDs and insert into a table variable
    INSERT INTO @StudentIDsList (ID)
    SELECT value
    FROM STRING_SPLIT(@StudentIDs, ',');

    -- Check if the number of questions provided matches the actual count
    IF @NumberOfStudents <> (SELECT COUNT(*) FROM @StudentIDsList)
    BEGIN
        PRINT 'Number of questions does not match the provided count.';
        RETURN;
    END

    -- Check if the provided Question IDs exist in the Questions table
    IF EXISTS (
        SELECT 1
        FROM @StudentIDsList Sl
        WHERE NOT EXISTS (
            SELECT 1
            FROM dbo.Students s
            WHERE s.Std_ID = sl.ID
        )
    )
    BEGIN
        PRINT 'One or more Question IDs do not exist in the Student table.';
        RETURN;
    END


    -- Insert data into Exam_Question table
    INSERT INTO [dbo].[Student_Exam] ([Student_ID],[Exam_ID],[Results])
    SELECT ID, @ExamID, '0'
    FROM @StudentIDsList;

    PRINT 'Data inserted into Student_Exam successfully.';
END;
GO
```

Uses

[dbo].[Student_Exam]
[dbo].[Students]
InstructorSC

[MangerSC].[CreateBranch]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	nvarchar(50)	100
@address	nvarchar(50)	100

SQL Script

```
CREATE PROC [MangerSC].[CreateBranch]
    @name nvarchar(50),
    @address nvarchar(50)

AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Perform the insertion
        INSERT INTO [dbo].[Branchs]
               ([Branch_name], [Branch_address] )
         VALUES
               (@name,@address );

        PRINT 'Branch Created successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while Created the Branch. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Branchs]

MangerSC

[MangerSC].[CreateCourses]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	nvarchar(50)	100
@min	int	4
@max	int	4
@decs	nvarchar(50)	100

SQL Script

```
CREATE PROC [MangerSC].[CreateCourses]
    @name NVARCHAR(50),
    @min int,
    @max int,
    @decs NVARCHAR(50)
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Perform the insertion
        INSERT INTO [dbo].[Courses]
               ([C_name], [C_minDegree] ,[C_maxDigree],  [C_Description])
         VALUES
               (@name, @min, @max, @decs);

        PRINT 'Courses created successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while creating the Courses. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
```

```
GO
```

Uses

[dbo].[Courses]

MangerSC

[MangerSC].[CreateInstructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	nvarchar(50)	100
@email	nvarchar(50)	100
@password	nvarchar(50)	100
@manager	int	4

SQL Script

```
CREATE PROC [MangerSC].[CreateInstructor]
    @name NVARCHAR(50),
    @email NVARCHAR(50),
    @password NVARCHAR(50),
    @manager INT
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Perform the insertion
        INSERT INTO [dbo].[Instructors]
               ([Inst_name],[Inst_email],[Inst_password] ,[manager_id] )
         VALUES
               (@name, @email, @password, @manager);

        PRINT 'Instructor created successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while creating the Instructor. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
```

```
GO
```

Uses

[dbo].[Instructors]

MangerSC

[MangerSC].[CreateInstructorINCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@IdCourses	int	4
@IdInstructor	int	4
@IdClass	int	4

SQL Script

```
CREATE PROC [MangerSC].[CreateInstructorINCourse]
    @IdCourses int,
    @IdInstructor int,
    @IdClass int

AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Perform the insertion
        INSERT INTO [dbo].[Inst_teach_course]
               ([course_ID],[Instructor_ID], [Class_ID],  [year])
         VALUES
               (@IdCourses, @IdInstructor, @IdClass, year(GETDATE()));

        PRINT 'Instructor added in Course successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while added the Instructor in Course. Error: ' + ERROR_-
MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Inst_teach_course]

MangerSC

[MangerSC].[CreateStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	nvarchar(50)	100
@email	nvarchar(50)	100
@password	nvarchar(50)	100
@intakeID	int	4
@branchID	int	4
@trackID	int	4
@classID	int	4

SQL Script

```
CREATE PROC [MangerSC].[CreateStudent]
    @name NVARCHAR(50),
    @email NVARCHAR(50),
    @password NVARCHAR(50),
    @intakeID INT,
    @branchID INT,
    @trackID INT,
    @classID INT
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Perform the insertion
        INSERT INTO [dbo].[Students]
               ([Std_name], [Std_email], [Std_password], [Intake_id], [barch_id], [track_id],
[class_id])
         VALUES
               (@name, @email, @password, @intakeID, @branchID, @trackID, @classID);

        PRINT 'Student created successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while creating the student. Error: ' + ERROR_MESSAGE();
    END CATCH
END;

GO
```

Uses

[dbo].[Students]
MangerSC

[MangerSC].[DeleteCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@id	int	4

SQL Script

```
CREATE PROC [MangerSC].[DeleteCourse]
    @id INT
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before deleting
        IF EXISTS (SELECT 1 FROM [dbo].[Courses] WHERE C_ID = @id)
        BEGIN
            -- Perform the deletion
            DELETE FROM [dbo].[Courses]
            WHERE C_ID = @id;

            PRINT 'Cousre deleted successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Cousre not found. No deletion performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while deleting the Cousre. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Courses]

MangerSC

[MangerSC].[DeleteInstructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@id	int	4

SQL Script

```
CREATE PROC [MangerSC].[DeleteInstructor]
    @id INT
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before deleting
        IF EXISTS (SELECT 1 FROM [dbo].[Instructors] WHERE Inst_ID = @id)
        BEGIN
            -- Perform the deletion
            DELETE FROM [dbo].[Instructors]
            WHERE Inst_ID = @id;

            PRINT 'Instructors deleted successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Instructors not found. No deletion performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while deleting the student. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Instructors]

MangerSC

[MangerSC].[DeleteStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@id	int	4

SQL Script

```
CREATE PROC [MangerSC].[DeleteStudent]
    @id INT
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before deleting
        IF EXISTS (SELECT 1 FROM Students WHERE Std_ID = @id)
        BEGIN
            -- Perform the deletion
            DELETE FROM Students
            WHERE Std_ID = @id;

            PRINT 'Student deleted successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Student not found. No deletion performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while deleting the student. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Students]

MangerSC

[MangerSC].[EditStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	nvarchar(50)	100
@email	nvarchar(50)	100
@id	int	4

SQL Script

```
CREATE PROCEDURE [MangerSC].[EditStudent]
    @name NVARCHAR(50),
    @email NVARCHAR(50),
    @id INT
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before updating
        IF EXISTS (SELECT 1 FROM Students WHERE Std_ID = @id)
        BEGIN
            -- Perform the update
            UPDATE [dbo].[Students]
            SET Std_name = @name, Std_email = @email
            WHERE Std_ID = @id;

            PRINT 'Student information updated successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Student not found. No update performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while updating the student information. Error: ' + ERROR_-
```

```
MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Students]
MangerSC

[MangerSC].[EidtBranch]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@address	nvarchar(50)	100
@id	int	4

SQL Script

```
CREATE PROC [MangerSC].[EidtBranch]
@address nvarchar(50),
@id int
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before updating
        IF EXISTS (SELECT 1 FROM [dbo].[Branchs] WHERE Branch_ID= @id)
        BEGIN
            -- Perform the update
            UPDATE [dbo].[Branchs]
             set  Branch_address = @address
            where Branch_ID = @id

            PRINT 'Branchs information updated successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Branchs not found. No update performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while updating the Branchs information. Error: ' + ERROR_-
MESSAGE();
    END CATCH
END;
```

```
GO
```

Uses

[dbo].[Branchs]

MangerSC

[MangerSC].[EidtCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@min	int	4
@max	int	4
@id	int	4

SQL Script

```
CREATE PROC [MangerSC].[EidtCourse]
@min int,
@max int,
@id int
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before updating
        IF EXISTS (SELECT 1 FROM Courses WHERE C_ID= @id)
        BEGIN
            -- Perform the update
            UPDATE Courses
 set  C_minDegree = @min , C_maxDigree = @max
where C_ID = @id

            PRINT 'Course information updated successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Course not found. No update performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while updating the Course information. Error: ' + ERROR_-
```

```
MESSAGE();
    END CATCH
END;

GO
```

Uses

[dbo].[Courses]
MangerSC

[MangerSC].[EidtInstForEachCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@newInstID	int	4
@oldInstID	int	4
@oldClassID	int	4
@oldCourseID	int	4

SQL Script

```
CREATE PROC [MangerSC].[EidtInstForEachCourse]
@newInstID int,
@oldInstID int,
@oldClassID int,
@oldCourseID int
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before updating
        IF EXISTS (SELECT 1 FROM Inst_teach_course WHERE course_ID= @oldCourseID and Class_ID =
@oldClassID and  Instructor_ID = @oldInstID)
        BEGIN
            -- Perform the update
            UPDATE Inst_teach_course
             set  Instructor_ID = @newInstID
            where course_ID= @oldCourseID and Class_ID = @oldClassID and  Instructor_ID = @old-
InstID

            PRINT 'Inst_teach_course information updated successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Inst_teach_course not found. No update performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while updating the Inst_teach_course information. Error: ' + 
ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Inst_teach_course]
MangerSC

[MangerSC].[EidtInstructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@email	nvarchar(50)	100
@password	nvarchar(50)	100
@id	int	4

SQL Script

```
CREATE PROC [MangerSC].[EidtInstructor]
@email nvarchar(50),
@password nvarchar(50),
@id int
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before updating
        IF EXISTS (SELECT 1 FROM [Instructors] WHERE Inst_ID = @id)
        BEGIN
            -- Perform the update
            UPDATE [dbo].[Instructors]
 set  Inst_email = @email , Inst_password = @password
where Inst_ID = @id

            PRINT 'Instructor information updated successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Instructor not found. No update performed.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors if any
        PRINT 'An error occurred while updating the Instructor information. Error: ' + ERROR_-
MESSAGE();
```

```
    END CATCH
END;
GO
```

Uses

[dbo].[Instructors]
MangerSC

[MangerSC].[EidtIntake_Depart]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@newDeptID	int	4
@oldDeptID	int	4
@id	int	4

SQL Script

```
CREATE PROC [MangerSC].[EidtIntake_Depart]
@newDeptID int,
@oldDeptID int,
@id int
AS
BEGIN
    SET NOCOUNT ON; -- This prevents the count of the number of rows affected from being returned

    BEGIN TRY
        -- Check if the student exists before updating
        IF EXISTS (SELECT 1 FROM [dbo].[Intake_Depart] WHERE [Intake_id]= @id and Depart_ID =
@oldDeptID)
        BEGIN
            -- Perform the update
            UPDATE [dbo].[Intake_Depart]
             set  Depart_ID = @newDeptID
            where [Intake_id] = @id and Depart_ID = @oldDeptID

            PRINT 'Intake_Depart information updated successfully.';
        END
        ELSE
        BEGIN
            PRINT 'Intake_Depart not found. No update performed.';
        END
    END TRY
    BEGIN CATCH
```

```
        -- Handle errors if any
        PRINT 'An error occurred while updating the Intake_Depart information. Error: ' + ERROR_-
MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Intake_Depart]
MangerSC

[StudentSC].[GetAvailableExams]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
CREATE PROC [StudentSC].[GetAvailableExams]
AS
BEGIN
    SET NOCOUNT ON;

    select * from [dbo].[GetAvailableExamsFun]()


END;
GO
```

Uses

[dbo].[GetAvailableExamsFun]
StudentSC

[StudentSC].[InsertAnswer]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StudentID	int	4
@ExamID	int	4
@QuestionID	int	4
@StudentAnswer	nvarchar(50)	100

SQL Script

```
CREATE PROCEDURE [StudentSC].[InsertAnswer]
    @StudentID INT,
    @ExamID INT,
    @QuestionID INT,
    @StudentAnswer NVARCHAR(50)
AS
BEGIN
    SET NOCOUNT ON;

    -- Check if the Student_ID and Exam_ID have a valid relationship
    IF EXISTS (
        SELECT 1
        FROM dbo.Student_Exam
        WHERE Student_ID = @StudentID
          AND Exam_ID = @ExamID
    )
    BEGIN
        -- Check if the exam is available for answering
        IF EXISTS (
            SELECT 1
            FROM dbo.GetAvailableExamsFun()
            WHERE E_ID = @ExamID
        )
        BEGIN
            -- Check if the Question_ID is associated with the Exam_ID
            IF EXISTS (
                SELECT 1
                FROM dbo.Exam_Question
                WHERE Exam_ID = @ExamID
                  AND Question_ID = @QuestionID
            )
            BEGIN
                -- Insert into Answer table
                INSERT INTO dbo.Answer (Student_ID, Exam_ID, Question_ID, Student_answer)
                VALUES (@StudentID, @ExamID, @QuestionID, @StudentAnswer);

            END
            ELSE
            BEGIN
                PRINT 'This question is not associated with the provided exam.';
            END;
        END
        ELSE
        BEGIN
            PRINT 'This exam is not currently available for answering.';
        END;
    END
    ELSE
    BEGIN
        PRINT 'This student is not allowed to this exam';
    END;
END;
GO
```

Uses

[dbo].[Answer]
[dbo].[Exam_Question]
[dbo].[Student_Exam]
[dbo].[GetAvailableExamsFun]
StudentSC

Table-valued Functions

Objects

Name
dbo.GetAvailableExamsFun

[dbo].[GetAvailableExamsFun]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
CREATE   FUNCTION [dbo].[GetAvailableExamsFun]()
RETURNS TABLE
AS
RETURN
(

SELECT Exam.E_ID , Questions.Q_Text, Questions.Q_type, Questions.Q_Mark, Exam.E_allownce
FROM    Exam_Question INNER JOIN
                  Exam ON Exam_Question.Exam_ID = Exam.E_ID INNER JOIN
                Questions ON Exam_Question.Question_ID = Questions.Q_ID
WHERE
        Exam.E_startTime <= CONVERT(TIME, GETDATE())
        AND Exam.E_endTime > CONVERT(TIME, GETDATE())
        AND Exam.E_Date = CONVERT(date, GETDATE())
);
GO
```

Uses

[dbo].[Exam]
[dbo].[Exam_Question]
[dbo].[Questions]

Used By

[StudentSC].[GetAvailableExams]
[StudentSC].[InsertAnswer]

Scalar-valued Functions

Objects

Name
dbo.CalculateTotalCorrectAnswers

[dbo].[CalculateTotalCorrectAnswers]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ExamID	int	4
@StudentID	int	4

SQL Script

```
CREATE   FUNCTION [dbo].[CalculateTotalCorrectAnswers](@ExamID INT, @StudentID INT)
RETURNS INT
AS
BEGIN
    DECLARE @TotalCorrectAnswers INT;

    SELECT @TotalCorrectAnswers = sum(q.Q_Mark)
    FROM dbo.Answer a
    INNER JOIN dbo.Questions q ON a.Question_ID = q.Q_ID
    WHERE a.Exam_ID = @ExamID
    AND a.Student_ID = @StudentID
    AND a.Student_answer = q.Q_correctAns;

    RETURN @TotalCorrectAnswers;
END;
GO
```

Uses

[dbo].[Answer]
[dbo].[Questions]

Used By

[dbo].[UpdateStudentExamResults]

Users

Objects

Name
instructor
manager
student

instructor

Properties

Property	Value
Type	SqlUser
Login Name	instructor
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [instructor] FOR LOGIN [instructor]
GO
```

manager

Properties

Property	Value
Type	SqlUser
Login Name	manager
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [manager] FOR LOGIN [manager]
GO
```

student

Properties

Property	Value
Type	SqlUser
Login Name	student
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [student] FOR LOGIN [student]
GO
```

Database Roles

Objects

Name
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
public

db_accessadmin

Properties

Property	Value
Owner	dbo

db_backupoperator

Properties

Property	Value
Owner	dbo

db_datareader

Properties

Property	Value
Owner	dbo

db_datawriter

Properties

Property	Value
Owner	dbo

db_ddladmin

Properties

Property	Value
Owner	dbo

db_denydatareader

Properties

Property	Value
Owner	dbo

db_denydatawriter

Properties

Property	Value
Owner	dbo

db_owner

Properties

Property	Value
Owner	dbo

db_securityadmin

Properties

Property	Value
Owner	dbo

public

Properties

Property	Value
Owner	dbo

Schemas

Objects

Name
InstructorSC
MangerSC
StudentSC

InstructorSC

Properties

Property	Value
Owner	instructor

SQL Script

```
CREATE SCHEMA [InstructorSC]
AUTHORIZATION [instructor]
GO
```

Used By

[InstructorSC].[ShowAllDataFromExam]
[InstructorSC].[ShowDataOFQuestionPool]
[InstructorSC].[ShowQuestionsInExam]
[InstructorSC].[ShowStudentInExam]
[InstructorSC].[ShowStudentsInClasses]
[InstructorSC].[CreateExam]
[InstructorSC].[CreateQuestion]
[InstructorSC].[DeleteQuestion]
[InstructorSC].[EditQuestions]
[InstructorSC].[EidtQuestionsMark]
[InstructorSC].[GetStudentByID]
[InstructorSC].[InsertExamQuestions]
[InstructorSC].[InsertExamQuestionsRandomly]
[InstructorSC].[insertStudentToExams]

MangerSC

Properties

Property	Value
Owner	manager

SQL Script

```
CREATE SCHEMA [MangerSC]
AUTHORIZATION [manager]
GO
```

Used By

[MangerSC].[ShowAllDataFromBranch]
[MangerSC].[ShowAllDataFromCourses]
[MangerSC].[ShowAllDataFromInstaructors]
[MangerSC].[ShowAllDataFromStudent]
[MangerSC].[ShowDepartmentInIntake]
[MangerSC].[ShowInstructorInCourseAndClass]
[MangerSC].[CreateBranch]
[MangerSC].[CreateCourses]
[MangerSC].[CreateInstructor]
[MangerSC].[CreateInstructorINCourse]
[MangerSC].[CreateStudent]
[MangerSC].[DeleteCourse]
[MangerSC].[DeleteInstructor]
[MangerSC].[DeleteStudent]
[MangerSC].[EditStudent]
[MangerSC].[EidtBranch]
[MangerSC].[EidtCourse]
[MangerSC].[EidtInstForEachCourse]
[MangerSC].[EidtInstructor]
[MangerSC].[EidtIntake_Depart]

StudentSC

Properties

Property	Value
Owner	student

SQL Script

```
CREATE SCHEMA [StudentSC]
AUTHORIZATION [student]
GO
```

Used By

[StudentSC].[ShowStudentResults]
[StudentSC].[GetAvailableExams]
[StudentSC].[InsertAnswer]