Online Examination System Documentation

Track: Mobile Cross Platform Intake44

Team Members:

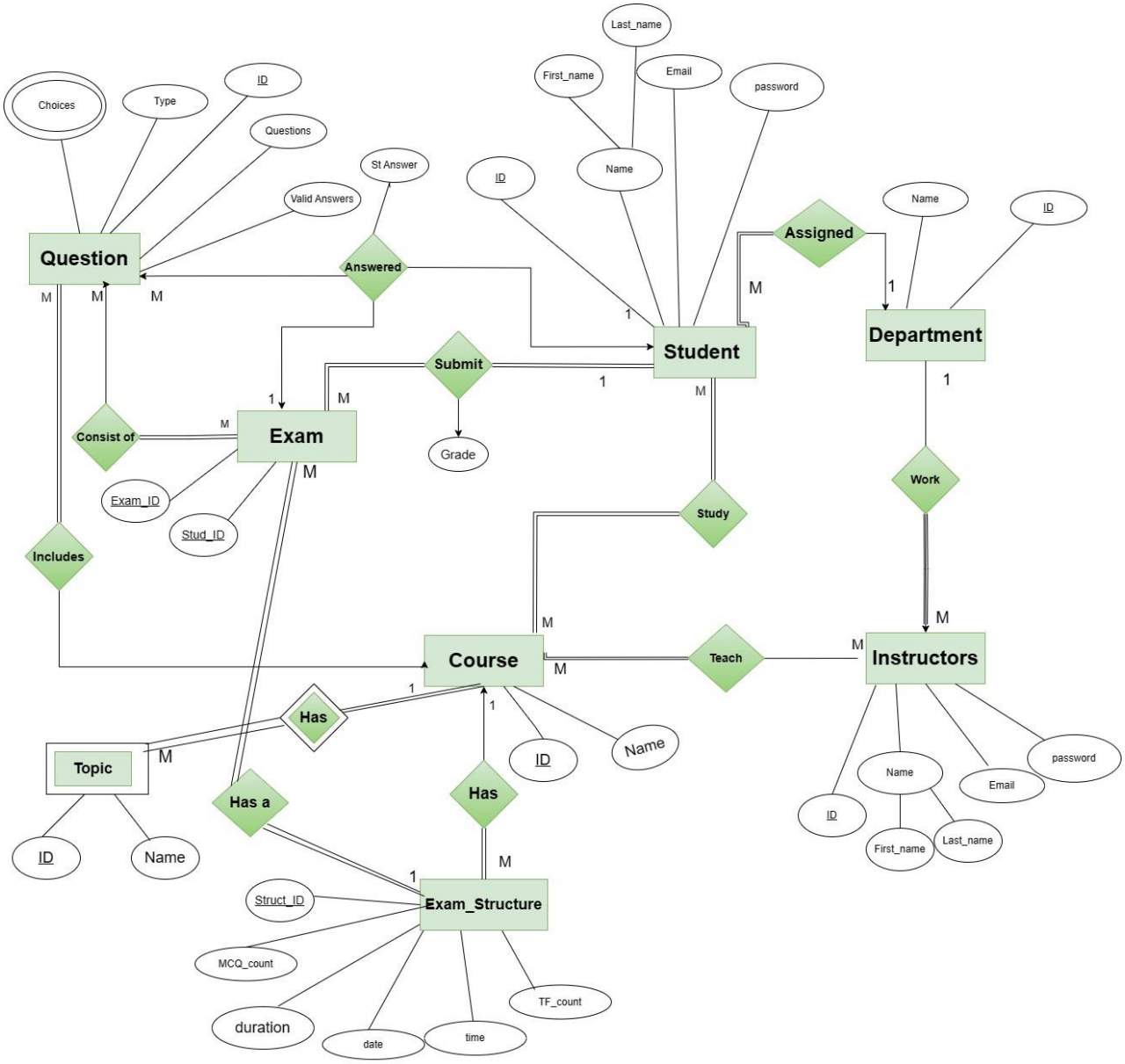
- **Shimaa Atef**
- **Khalid Elsayed**
- **Ranaa Essam**
- **Mazen Ayman**
- **Salah El-dein Nageh**

Under The Supervision of:

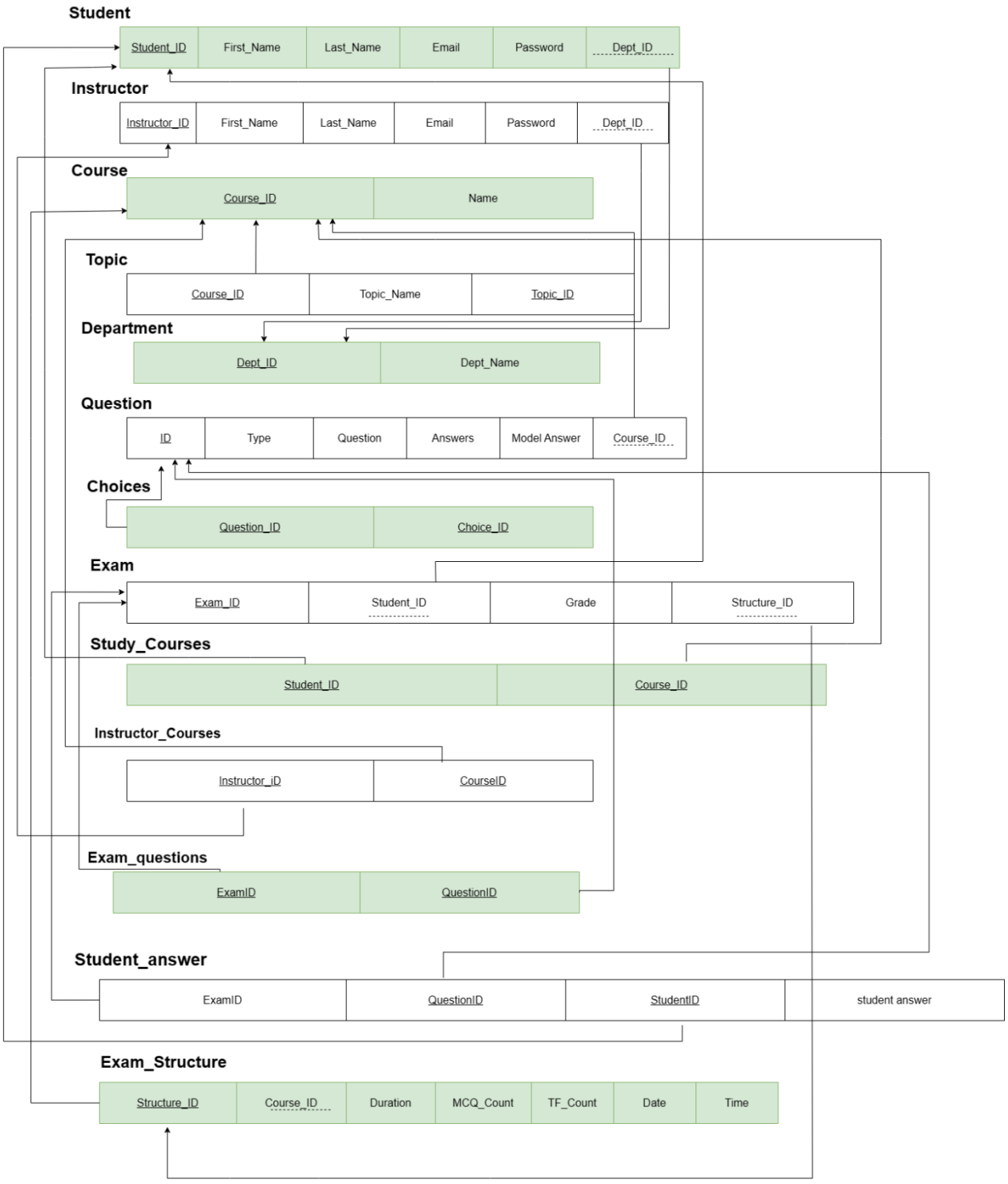
Dr. Rami Nagi

Table of Contents

ERD Diagram:


Last_name
First_name
Email
password
Choices
Type
ID
Questions
St Answer
Valid Answers
Name
ID
Name
ID
Assigned
Question
Answered
Student
Department
Submit
Grade
Consist of
Exam
Exam_ID
Stud_ID
Includes
Study
Work
Course
Teach
Instructors
Has
Topic
ID
Name
Has a
Has
ID
Name
Struct_ID
Exam_Structure
MCQ_count
TF_count
duration
date
time
ID
Name
Email
password
First_name
Last_name

Mapping:


Student
Student_ID
First_Name
Last_Name
Email
Password
Dept_ID
Instructor
Instructor_ID
First_Name
Last_Name
Email
Password
Dept_ID
Course
Course_ID
Name
Topic
Course_ID
Topic_Name
Topic_ID
Department
Dept_ID
Dept_Name
Question
ID
Type
Question
Answers
Model Answer
Course_ID
Choices
Question_ID
Choice_ID
Exam
Exam_ID
Student_ID
Grade
Structure_ID
Study_Courses
Student_ID
Course_ID
Instructor_Courses
Instructor_iD
CourseID
Exam_questions
ExamID
QuestionID
Student_answer
ExamID
QuestionID
StudentID
student answer
Exam_Structure
Structure_ID
Course_ID
Duration
MCQ_Count
TF_Count
Date
Time

DESKTOP-7IK5157\SQLEXPRESS

Databases (1)

- ExaminationSystem

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 (22631)
Physical Memory	6061
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

ExaminationSystem Database

Database Properties

Property	Value
SQL Server Version	Max
Compatibility Level	Max
Last backup time	02/27/2024
Last log backup time	-
Creation date	Jan 24 2024
Users	4
Database Encryption Enabled	False
Database Encryption Algorithm	None
Database size	80.00 MB
Unallocated space	57.77 MB

Database Options

Files

Name	Type	Size	Maxsize	Autogrowth	File Name
ExaminationSystem	Data	72.00 MB	unlimited	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\ExaminationSystem.mdf
ExaminationSystem_log	Log	8.00 MB	2048.00 GB	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\ExaminationSystem_log.ldf

Tables

Objects

Name
dbo.Choices
dbo.Course
dbo.Department
dbo.Exam
dbo.Exam_Question
dbo.Exam_Structure
dbo.Instructor
dbo.Instructor_Courses
dbo.Question
dbo.Student
dbo.Student_Answer
dbo.Student_Course
dbo.Topic

[dbo].[Choices]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	180
Created	07:37:24 22-1-2024
Last Modified	08:21:11 22-1-2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK C FK	Question_ID	int	4	NOT NULL
PK C	Choices	varchar(200)	200	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Choices	Question_ID, Choices	True

Foreign Keys

Name	Update	Delete	Columns
FK_Choices_Question	Cascade	Cascade	Question_ID->[dbo].[Question].[Question_ID]

SQL Script

```
CREATE TABLE [dbo].[Choices]
(
[Question_ID] [int] NOT NULL,
[Choices] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Choices] ADD CONSTRAINT [PK_Choices] PRIMARY KEY CLUSTERED
([Question_ID], [Choices]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Choices] ADD CONSTRAINT [FK_Choices_Question] FOREIGN KEY
([Question_ID]) REFERENCES [dbo].[Question] ([Question_ID]) ON DELETE CASCADE ON UPDATE
CASCADE
```

```
GO
```

Uses

[dbo].[Question]

Used By

[dbo].[Delete_Choices]
[dbo].[Delete_Course]
[dbo].[Delete_Question]
[dbo].[Get_Exam_With_Choices_Report]
[dbo].[Insert_Choices]
[dbo].[Insert_Question]
[dbo].[Select_Choices]
[dbo].[Update_Choices]

[dbo].[Course]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	5
Created	02:39:59 2024 ,يناير 19 م
Last Modified	01:27:32 2024 ,فبراير 16 ص

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK C	Course_ID	int	4	NOT NULL
	Crs_Name	varchar(30)	30	NULL allowed

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Course	Course_ID	True

SQL Script

```
CREATE TABLE [dbo].[Course]
(
[Course_ID] [int] NOT NULL,
[Crs_Name] [varchar] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Course] ADD CONSTRAINT [PK_Course] PRIMARY KEY CLUSTERED ([Course_-
ID]) ON [PRIMARY]
GO
```

Used By

[dbo].[Exam_Structure]
[dbo].[Instructor_Courses]
[dbo].[Question]
[dbo].[Student_Course]
[dbo].[Topic]

[dbo].[Delete_Course]
[dbo].[GetInstCrsStuds_Report]
[dbo].[GetStudentGradesReport]
[dbo].[GetTopicNameReport]
[dbo].[Insert_Course]
[dbo].[Select_All_Course_With_Topics]
[dbo].[Select_Course]
[dbo].[Select_Course_With_Topics]
[dbo].[Select_InstructorCourse]
[dbo].[Select_Question]
[dbo].[Select_Student_Course]
[dbo].[Select_Topic]
[dbo].[SelectStudentExams]
[dbo].[Update_Course]

[dbo].[Department]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	1
Created	07:10:50 2024 ,يناير 22 م
Last Modified	07:21:02 2024 ,يناير 22 م

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Dept_ID	int	4	NOT NULL	1 - 1
	Dept_Name	varchar(20)	20	NULL allowed	

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] [varchar] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Department] ADD CONSTRAINT [PK_Department] PRIMARY KEY CLUSTERED
([Dept_ID]) ON [PRIMARY]
GO
```

Used By

[dbo].[Instructor]
[dbo].[Student]
[dbo].[Delete_Department]
[dbo].[GetStudentInformationReport]
[dbo].[Insert_Department]

[dbo].[Select_Department]

[dbo].[Select_InstructorInfo]

[dbo].[Select_Student]

[dbo].[Update_Department]

[dbo].[Exam]

Properties

Property	Value
Row Count (~)	3
Created	12:54:51 2024 ,ص 16 فبراير
Last Modified	01:27:32 2024 ,ص 16 فبراير

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Exam_ID	int	4	NOT NULL	1 - 1
FK	Student_ID	int	4	NOT NULL	
	Grade	int	4	NULL allowed	
FK	Structure_ID	int	4	NOT NULL	

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Exam	Exam_ID	True

Foreign Keys

Name	Columns
FK_Exam_Exam_Structure	Structure_ID->[dbo].[Exam_Structure].[Structure_ID]
FK_Exam_Student	Student_ID->[dbo].[Student].[Student_ID]

SQL Script

```
CREATE TABLE [dbo].[Exam]
(
[Exam_ID] [int] NOT NULL IDENTITY(1, 1),
[Student_ID] [int] NOT NULL,
[Grade] [int] NULL,
[Structure_ID] [int] NOT NULL
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [PK_Exam] PRIMARY KEY CLUSTERED ([Exam_ID]) ON
[PRIMARY]
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [FK_Exam_Exam_Structure] FOREIGN KEY
([Structure_ID]) REFERENCES [dbo].[Exam_Structure] ([Structure_ID])
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [FK_Exam_Student] FOREIGN KEY ([Student_ID])
REFERENCES [dbo].[Student] ([Student_ID])
GO
```

Uses

[dbo].[Exam_Structure]
[dbo].[Student]

Used By

[dbo].[Exam_Question]
[dbo].[Student_Answer]
[dbo].[Create_Exams_For_Department]
[dbo].[Delete_Course]
[dbo].[Delete_Exam]
[dbo].[ExamCorrection]
[dbo].[ExamGeneration]
[dbo].[GetExamDetails]
[dbo].[GetStudentGradesReport]
[dbo].[Select_Exams_By_Course]
[dbo].[SelectStudentExams]
[dbo].[Update_Exam_Grade]

[dbo].[Exam_Question]

Properties

Property	Value
Row Count (~)	20
Created	02:57:13 2024 ,19 يناير م
Last Modified	12:54:52 2024 ,16 فبراير ص

Columns

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

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Exam_Question	Exam_ID, Question_ID	True

Foreign Keys

Name	Columns
FK_Exam_Question_Exam	Exam_ID->[dbo].[Exam].[Exam_ID]
FK_Exam_Question_Question	Question_ID->[dbo].[Question].[Question_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] ([Exam_ID])
GO
ALTER TABLE [dbo].[Exam_Question] ADD CONSTRAINT [FK_Exam_Question_Question] FOREIGN
KEY ([Question_ID]) REFERENCES [dbo].[Question] ([Question_ID])
GO
```

Uses

[dbo].[Exam]
[dbo].[Question]

Used By

[dbo].[Delete_Exam]
[dbo].[Delete_Exam_Question]
[dbo].[Delete_Question]
[dbo].[ExamAnswers]
[dbo].[ExamGeneration]
[dbo].[Get_Exam_With_Choices_Report]
[dbo].[GetQuestionsWithStudentAnswersReport]
[dbo].[Insert_Exam_Question]
[dbo].[Select_Exam_Question]
[dbo].[Update_Exam_Question]

[dbo].[Exam_Structure]

Properties

Property	Value
Row Count (~)	61
Created	01:27:31 2024 ,16 فبراير ص
Last Modified	01:27:32 2024 ,16 فبراير ص

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Structure_ID	int	4	NOT NULL	1 - 1
FK	Course_ID	int	4	NULL allowed	
	Duration	int	4	NULL allowed	
	MCQ_Count	int	4	NULL allowed	
	TF_Count	int	4	NULL allowed	
	Date	date	3	NULL allowed	
	time	time	5	NULL allowed	

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Exam_Structure	Structure_ID	True

Foreign Keys

Name	Columns
FK_Exam_Structure_Course	Course_ID->[dbo].[Course].[Course_ID]

SQL Script

```
CREATE TABLE [dbo].[Exam_Structure]
(
[Structure_ID] [int] NOT NULL IDENTITY(1, 1),
[Course_ID] [int] NULL,
[Duration] [int] NULL,
[MCQ_Count] [int] NULL,
```



```
[TF_Count] [int] NULL,
[Date] [date] NULL,
[time] [time] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam_Structure] ADD CONSTRAINT [PK_Exam_Structure] PRIMARY KEY
CLUSTERED ([Structure_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam_Structure] ADD CONSTRAINT [FK_Exam_Structure_Course] FOREIGN
KEY ([Course_ID]) REFERENCES [dbo].[Course] ([Course_ID])
GO
```

Uses

[dbo].[Course]

Used By

[dbo].[Exam]
[dbo].[Create_Exams_For_Department]
[dbo].[ExamGeneration]
[dbo].[GetExamDetails]
[dbo].[GetStudentGradesReport]
[dbo].[Select_Exams_By_Course]
[dbo].[SelectStudentExams]

[dbo].[Instructor]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	1
Created	07:21:02 2024 ,يناير 22 م
Last Modified	07:23:26 2024 ,يناير 22 م

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Instructor_ID	int	4	NOT NULL	1 - 1
	First_Name	varchar(15)	15	NOT NULL	
	Second_Name	varchar(15)	15	NOT NULL	
	Email	varchar(50)	50	NOT NULL	
	Password	varchar(50)	50	NOT NULL	
FK	Dept_ID	int	4	NULL allowed	

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Instructor	Instructor_ID	True

Foreign Keys

Name	Columns
FK_Instructor_Department	Dept_ID->[dbo].[Department].[Dept_ID]

SQL Script

```
CREATE TABLE [dbo].[Instructor]
(
[Instructor_ID] [int] NOT NULL IDENTITY(1, 1),
[First_Name] [varchar] (15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Second_Name] [varchar] (15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Email] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
```



```
[Password] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Dept_ID] [int] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Instructor] ADD CONSTRAINT [PK_Instructor] PRIMARY KEY CLUSTERED
([Instructor_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Instructor] ADD CONSTRAINT [FK_Instructor_Department] FOREIGN KEY
([Dept_ID]) REFERENCES [dbo].[Department] ([Dept_ID])
GO
```

Uses

[dbo].[Department]

Used By

[dbo].[Instructor_Courses]
[dbo].[Delete_Department]
[dbo].[Delete_Instructor]
[dbo].[Insert_Instructor]
[dbo].[Select_InstructorCourse]
[dbo].[Select_InstructorInfo]
[dbo].[Update_Instructor]

[dbo].[Instructor_Courses]

Properties

Property	Value
Row Count (~)	1
Created	02:49:00 2024 ,يناير 19 م
Last Modified	07:23:26 2024 ,يناير 22 م

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK C FK	Instructor_ID	int	4	NOT NULL
PK C FK	Course_ID	int	4	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Instructor_Courses	Instructor_ID, Course_ID	True

Foreign Keys

Name	Update	Delete	Columns
FK_Instructor_Courses_Course	Cascade	Cascade	Course_ID->[dbo].[Course].[Course_ID]
FK_Instructor_Courses_Instructor	Cascade	Cascade	Instructor_ID->[dbo].[Instructor].[Instructor_ID]

SQL Script

```
CREATE TABLE [dbo].[Instructor_Courses]
(
[Instructor_ID] [int] NOT NULL,
[Course_ID] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Instructor_Courses] ADD CONSTRAINT [PK_Instructor_Courses] PRIMARY
KEY CLUSTERED ([Instructor_ID], [Course_ID]) ON [PRIMARY]
GO
```



```
ALTER TABLE [dbo].[Instructor_Courses] ADD CONSTRAINT [FK_Instructor_Courses_Course]
FOREIGN KEY ([Course_ID]) REFERENCES [dbo].[Course] ([Course_ID]) ON DELETE CASCADE ON
UPDATE CASCADE
GO
ALTER TABLE [dbo].[Instructor_Courses] ADD CONSTRAINT [FK_Instructor_Courses_-
Instructor] FOREIGN KEY ([Instructor_ID]) REFERENCES [dbo].[Instructor] ([Instructor_-
ID]) ON DELETE CASCADE ON UPDATE CASCADE
GO
```

Uses

[dbo].[Course]
[dbo].[Instructor]

Used By

[dbo].[Delete_Course]
[dbo].[Delete_Instructor]
[dbo].[Delete_Instructor_Course]
[dbo].[GetInstCrsStuds_Report]
[dbo].[Insert_Instructor_Course]
[dbo].[Select_InstructorCourse]
[dbo].[Update_Instructor_Course]

[dbo].[Question]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	90
Created	08:21:11 2024 ,يناير 25 م
Last Modified	08:21:11 2024 ,يناير 25 م

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Question_ID	int	4	NOT NULL	11 - 1
	Type	varchar(3)	3	NOT NULL	
	Question	varchar(200)	200	NOT NULL	
	Model_Answer	varchar(200)	200	NOT NULL	
FK	Course_ID	int	4	NOT NULL	

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Question	Question_ID	True

Foreign Keys

Name	Columns
FK_Question_Course	Course_ID->[dbo].[Course].[Course_ID]

SQL Script

```
CREATE TABLE [dbo].[Question]
(
[Question_ID] [int] NOT NULL IDENTITY(11, 1),
[Type] [varchar] (3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Question] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Model_Answer] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Course_ID] [int] NOT NULL
```

```
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [PK_Question] PRIMARY KEY CLUSTERED
([Question_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [FK_Question_Course] FOREIGN KEY ([Course_-
ID]) REFERENCES [dbo].[Course] ([Course_ID])
GO
```

Uses

[dbo].[Course]

Used By

[dbo].[Choices]
[dbo].[Exam_Question]
[dbo].[Student_Answer]
[dbo].[Delete_Course]
[dbo].[Delete_Question]
[dbo].[ExamCorrection]
[dbo].[ExamGeneration]
[dbo].[Get_Exam_With_Choices_Report]
[dbo].[GetQuestionsWithStudentAnswersReport]
[dbo].[Insert_Question]
[dbo].[Select_Exam_Question]
[dbo].[Select_Question]
[dbo].[Update_Question]

[dbo].[Student]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	3
Created	07:05:07 2024 ,22 يناير م
Last Modified	12:54:52 2024 ,16 فبراير ص

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity	Default
PK C	Student_ID	int	4	NOT NULL	1 - 1	
	First_Name	varchar(15)	15	NOT NULL		(' ')
	Second_Name	varchar(15)	15	NOT NULL		(' ')
	Email	varchar(50)	50	NOT NULL		
	Password	varchar(50)	50	NOT NULL		
FK	Dept_ID	int	4	NULL allowed		

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Student	Student_ID	True

Foreign Keys

Name	Columns
FK_Student_Department	Dept_ID->[dbo].[Department].[Dept_ID]

SQL Script

```
CREATE TABLE [dbo].[Student]
(
[Student_ID] [int] NOT NULL IDENTITY(1, 1),
[First_Name] [varchar] (15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL CONSTRAINT
[DF_Student_First_Name] DEFAULT (' '),
[Second_Name] [varchar] (15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL CONSTRAINT
[DF_Student_Second_Name] DEFAULT (' '),
```



```
[Email] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Password] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Dept_ID] [int] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [PK_Student] PRIMARY KEY CLUSTERED
([Student_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [FK_Student_Department] FOREIGN KEY ([Dept_-
ID]) REFERENCES [dbo].[Department] ([Dept_ID])
GO
```

Uses

[dbo].[Department]

Used By

[dbo].[Exam]
[dbo].[Student_Answer]
[dbo].[Student_Course]
[dbo].[Create_Exams_For_Department]
[dbo].[Delete_Department]
[dbo].[Delete_Student]
[dbo].[GetStudentInformationReport]
[dbo].[Insert_Student]
[dbo].[Select_Student]
[dbo].[Select_Student_Course]
[dbo].[Update_Student]

[dbo].[Student_Answer]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	10
Created	07:37:23 2024 ,يناير 22 م
Last Modified	12:54:52 2024 ,فبراير 16 ص

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
FK	Exam_ID	int	4	NOT NULL
PK C FK	Question_ID	int	4	NOT NULL
PK C FK	Student_ID	int	4	NOT NULL
	Student_Answer	varchar(200)	200	NULL allowed

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Student_Answer	Question_ID, Student_ID	True

Foreign Keys

Name	Columns
FK_Student_Answer_Exam	Exam_ID->[dbo].[Exam].[Exam_ID]
FK_Student_Answer_Question	Question_ID->[dbo].[Question].[Question_ID]
FK_Student_Answer_Student	Student_ID->[dbo].[Student].[Student_ID]

SQL Script

```
CREATE TABLE [dbo].[Student_Answer]
(
[Exam_ID] [int] NOT NULL,
[Question_ID] [int] NOT NULL,
```



```
[Student_ID] [int] NOT NULL,
[Student_Answer] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Answer] ADD CONSTRAINT [PK_Student_Answer] PRIMARY KEY
CLUSTERED ([Question_ID], [Student_ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Answer] ADD CONSTRAINT [FK_Student_Answer_Exam] FOREIGN KEY
([Exam_ID]) REFERENCES [dbo].[Exam] ([Exam_ID])
GO
ALTER TABLE [dbo].[Student_Answer] ADD CONSTRAINT [FK_Student_Answer_Question] FOREIGN
KEY ([Question_ID]) REFERENCES [dbo].[Question] ([Question_ID])
GO
ALTER TABLE [dbo].[Student_Answer] ADD CONSTRAINT [FK_Student_Answer_Student] FOREIGN
KEY ([Student_ID]) REFERENCES [dbo].[Student] ([Student_ID])
GO
```

Uses

[dbo].[Exam]
[dbo].[Question]
[dbo].[Student]

Used By

[dbo].[Delete_Course]
[dbo].[Delete_Student_Answer]
[dbo].[ExamAnswers]
[dbo].[ExamCorrection]
[dbo].[GetQuestionsWithStudentAnswersReport]
[dbo].[Insert_Student_Answer]
[dbo].[Select_Student_Answer]
[dbo].[Update_Student_Answer]

[dbo].[Student_Course]

Properties

Property	Value
Row Count (~)	3
Created	02:47:26 2024 ,يناير 19 م
Last Modified	07:19:34 2024 ,يناير 22 م

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK C FK	Student_ID	int	4	NOT NULL
PK C FK	Course_ID	int	4	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Student_Course	Student_ID, Course_ID	True

Foreign Keys

Name	Update	Delete	Columns
FK_Student_Course_Course	Cascade	Cascade	Course_ID->[dbo].[Course].[Course_ID]
FK_Student_Course_Student	Cascade	Cascade	Student_ID->[dbo].[Student].[Student_ID]

SQL Script

```
CREATE TABLE [dbo].[Student_Course]
(
[Student_ID] [int] NOT NULL,
[Course_ID] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Course] ADD CONSTRAINT [PK_Student_Course] PRIMARY KEY
CLUSTERED ([Student_ID], [Course_ID]) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[Student_Course] ADD CONSTRAINT [FK_Student_Course_Course] FOREIGN
KEY ([Course_ID]) REFERENCES [dbo].[Course] ([Course_ID]) ON DELETE CASCADE ON UPDATE
CASCADE
GO
ALTER TABLE [dbo].[Student_Course] ADD CONSTRAINT [FK_Student_Course_Student] FOREIGN
KEY ([Student_ID]) REFERENCES [dbo].[Student] ([Student_ID]) ON DELETE CASCADE ON
UPDATE CASCADE
GO
```

Uses

[dbo].[Course]
[dbo].[Student]

Used By

[dbo].[Delete_Course]
[dbo].[Delete_Student_Course]
[dbo].[GetInstCrsStuds_Report]
[dbo].[Insert_Student_Course]
[dbo].[Select_Student_Course]

[dbo].[Topic]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	0
Created	02:33:57 2024 ,يناير 19 م
Last Modified	07:14:59 2024 ,يناير 22 م

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK	Course_ID	int	4	NOT NULL
PK	Topic_Name	varchar(50)	50	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK	PK_Topic	Course_ID, Topic_Name	True

Foreign Keys

Name	Update	Delete	Columns
FK_Topic_Course	Cascade	Cascade	Course_ID->[dbo].[Course].[Course_ID]

SQL Script

```
CREATE TABLE [dbo].[Topic]
(
[Course_ID] [int] NOT NULL,
[Topic_Name] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Topic] ADD CONSTRAINT [PK_Topic] PRIMARY KEY CLUSTERED ([Course_ID],
[Topic_Name]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Topic] ADD CONSTRAINT [FK_Topic_Course] FOREIGN KEY ([Course_ID])
REFERENCES [dbo].[Course] ([Course_ID]) ON DELETE CASCADE ON UPDATE CASCADE
```

```
GO
```

Uses

[dbo].[Course]

Used By

[dbo].[Delete_Course]
[dbo].[Delete_Topic]
[dbo].[GetTopicNameReport]
[dbo].[Insert_Topic]
[dbo].[Select_All_Course_With_Topics]
[dbo].[Select_Course_With_Topics]
[dbo].[Select_Topic]
[dbo].[Select_TopicNameById]
[dbo].[Select_TopicNames]
[dbo].[Update_Topic_Name]

Stored Procedures

Objects

Name
dbo.Create_Exams_For_Department
dbo.DataDictionary
dbo.DataDictionary1
dbo.Delete_Choices
dbo.Delete_Course
dbo.Delete_Department
dbo.Delete_Exam
dbo.Delete_Exam_Question
dbo.Delete_Instructor
dbo.Delete_Instructor_Course
dbo.Delete_Question
dbo.Delete_Student
dbo.Delete_Student_Answer
dbo.Delete_Student_Course
dbo.Delete_Topic
dbo.ExamAnswers
dbo.ExamCorrection
dbo.ExamGeneration
dbo.Get_Exam_With_Choices_Report
dbo.GetExamDetails
dbo.GetInstCrsStuds_Report
dbo.GetQuestionsWithStudentAnswersReport
dbo.GetStudentGradesReport
dbo.GetStudentInformationReport
dbo.GetTopicNameReport
dbo.Insert_Choices
dbo.Insert_Course
dbo.Insert_Department
dbo.Insert_Exam_Question
dbo.Insert_Instructor
dbo.Insert_Instructor_Course
dbo.Insert_Question
dbo.Insert_Student
dbo.Insert_Student_Answer

dbo.Insert_Student_Course
dbo.Insert_Topic
dbo.Select_All_Course_With_Topics
dbo.Select_Choices
dbo.Select_Course
dbo.Select_Course_With_Topics
dbo.Select_Department
dbo.Select_Exam_Question
dbo.Select_Exams_By_Course
dbo.Select_InstructorCourse
dbo.Select_InstructorInfo
dbo.Select_Question
dbo.Select_Student
dbo.Select_Student_Answer
dbo.Select_Student_Course
dbo.Select_Topic
dbo.Select_TopicNameById
dbo.Select_TopicNames
dbo.SelectStudentExams
dbo.Update_Choices
dbo.Update_Course
dbo.Update_Department
dbo.Update_Exam_Grade
dbo.Update_Exam_Question
dbo.Update_Instructor
dbo.Update_Instructor_Course
dbo.Update_Question
dbo.Update_Student
dbo.Update_Student_Answer
dbo.Update_Topic_Name

[dbo].[Create_Exams_For_Department]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@DepartmentID	int	4
@CourseID	int	4
@Duration	int	4
@NumsOfMCQ	int	4
@NumsOfTF	int	4
@ExamDate	date	3
@ExamTime	time	5

SQL Script

```
CREATE PROCEDURE [dbo].[Create_Exams_For_Department]
    @DepartmentID INT,
    @CourseID INT,
    @Duration INT,
    @NumsOfMCQ INT,
    @NumsOfTF INT,
    @ExamDate DATE,
    @ExamTime TIME
AS
BEGIN
    DECLARE @StructureID INT;

     IF @NumsOfMCQ+@NumsOfTF!= 10
    BEGIN
        -- Total number of questions is not 10, set message
        SELECT 'Total number of questions (MCQ + TF) must be 10.' AS Message;
        RETURN;
    END

    -- Check if exams already exist for the specified course in the department
    IF EXISTS (SELECT 1 FROM Exam_Structure es
```



```
                INNER JOIN Exam e ON es.Structure_ID = e.Structure_ID
                INNER JOIN Student s ON e.Student_ID = s.Student_ID
                WHERE es.Course_ID = @CourseID
                  AND s.Dept_ID = @DepartmentID)
    BEGIN
        -- Exams already exist, set message
        SELECT 'Exams already exist for the specified course in the department.' AS
Message;
        RETURN;
    END

    -- Insert into Exam_Structure
    INSERT INTO Exam_Structure (Course_ID, Duration, MCQ_Count, TF_Count, Date, Time)
    VALUES (@CourseID, @Duration, @NumsOfMCQ, @NumsOfTF, @ExamDate, @ExamTime);

    -- Get the Structure_ID of the inserted record
    SET @StructureID = SCOPE_IDENTITY();

    -- Insert exams for all students in the specified department
    INSERT INTO Exam (Student_ID, Grade, Structure_ID)
    SELECT
        Student.Student_ID,
        NULL AS Grade, -- Assuming grade is not specified in this scenario
        @StructureID AS Structure_ID
    FROM
        Student
    WHERE
        Student.Dept_ID = @DepartmentID;

    -- Exams inserted successfully, set message
    SELECT 'Exams created successfully.' AS Message;
END
GO
```

Uses

[dbo].[Exam]
[dbo].[Exam_Structure]
[dbo].[Student]

[dbo].[DataDictionary]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@tableName	varchar(128)	128

SQL Script

```
CREATE proc [dbo].[DataDictionary]
    @tableName VARCHAR(128) as
    begin
SELECT
    schemas.name AS SchemaName,
    tables.name AS TableName,
    columns.name AS ColumnName,
    types.name AS DataTypeName,
    indexes.name AS PrimaryKey,
    indexes.is_primary_key,
    fk.name AS ForeignKeyName,
    ref_tables.name AS ReferencedTableName,
    ISNULL(ep.value, '') AS TableDescription
FROM sys.tables
INNER JOIN sys.columns
    ON tables.object_id = columns.object_id
INNER JOIN sys.types
    ON types.user_type_id = columns.user_type_id
INNER JOIN sys.schemas
    ON schemas.schema_id = tables.schema_id
LEFT JOIN sys.index_columns
    ON columns.object_id = index_columns.object_id AND columns.column_id =
index_columns.column_id
LEFT JOIN sys.indexes
    ON index_columns.object_id = indexes.object_id AND index_columns.index_id =
indexes.index_id
LEFT JOIN sys.foreign_key_columns
    ON columns.object_id = foreign_key_columns.parent_object_id AND columns.column_id =
foreign_key_columns.parent_column_id
LEFT JOIN sys.foreign_keys AS fk
```



```
    ON foreign_key_columns.constraint_object_id = fk.object_id
LEFT JOIN sys.tables AS ref_tables
    ON fk.referenced_object_id = ref_tables.object_id
LEFT JOIN sys.extended_properties AS ep
    ON tables.object_id = ep.major_id AND ep.minor_id = 0 AND ep.class = 1
WHERE tables.name= @tableName
ORDER BY TableName, ColumnName;
end
GO
```


[dbo].[DataDictionary1]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@tableName	varchar(128)	128

SQL Script

```
 CREATE    proc [dbo].[DataDictionary1]
    @tableName VARCHAR(128) as
    begin
SELECT
Distinct
    schemas.name AS SchemaName,
     tables.name AS TableName,
      indexes.name AS PrimaryKey,
       ISNULL(ep.value, '') AS TableDescription

FROM sys.tables
INNER JOIN sys.columns
    ON tables.object_id = columns.object_id
INNER JOIN sys.types
    ON types.user_type_id = columns.user_type_id
INNER JOIN sys.schemas
    ON schemas.schema_id = tables.schema_id
LEFT JOIN sys.index_columns
    ON columns.object_id = index_columns.object_id AND columns.column_id =
index_columns.column_id
LEFT JOIN sys.indexes
    ON index_columns.object_id = indexes.object_id AND index_columns.index_id =
indexes.index_id
LEFT JOIN sys.foreign_key_columns
    ON columns.object_id = foreign_key_columns.parent_object_id AND columns.column_id =
foreign_key_columns.parent_column_id
LEFT JOIN sys.foreign_keys AS fk
    ON foreign_key_columns.constraint_object_id = fk.object_id
LEFT JOIN sys.tables AS ref_tables
```



```
    ON fk.referenced_object_id = ref_tables.object_id
LEFT JOIN sys.extended_properties AS ep
    ON tables.object_id = ep.major_id AND ep.minor_id = 0 AND ep.class = 1
WHERE tables.name= @tableName and   indexes.name  is not NULL



SELECT


    columns.name AS ColumnName,
    types.name AS DataTypeName,

    indexes.is_primary_key,
    fk.name AS ForeignKeyName,
    ref_tables.name AS ReferencedTableName

FROM sys.tables
INNER JOIN sys.columns
    ON tables.object_id = columns.object_id
INNER JOIN sys.types
    ON types.user_type_id = columns.user_type_id
INNER JOIN sys.schemas
    ON schemas.schema_id = tables.schema_id
LEFT JOIN sys.index_columns
    ON columns.object_id = index_columns.object_id AND columns.column_id =
index_columns.column_id
LEFT JOIN sys.indexes
    ON index_columns.object_id = indexes.object_id AND index_columns.index_id =
indexes.index_id
LEFT JOIN sys.foreign_key_columns
    ON columns.object_id = foreign_key_columns.parent_object_id AND columns.column_id =
foreign_key_columns.parent_column_id
LEFT JOIN sys.foreign_keys AS fk
    ON foreign_key_columns.constraint_object_id = fk.object_id
LEFT JOIN sys.tables AS ref_tables
    ON fk.referenced_object_id = ref_tables.object_id
LEFT JOIN sys.extended_properties AS ep
    ON tables.object_id = ep.major_id AND ep.minor_id = 0 AND ep.class = 1
WHERE tables.name= @tableName
ORDER BY tables.name,  columns.name;

end

GO
```


[dbo].[Delete_Choices]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--4-Delete choices
create proc [dbo].[Delete_Choices] @qid int
as
    begin
    set nocount on ;
    if exists(select c.Question_ID  from Choices c where c.Question_ID=@qid)
    begin
    begin try

    --deleting the answers
    delete from Choices
    where Choices.Question_ID=@qid
    end try
    begin catch
    select ERROR_NUMBER() AS ErrorNumber,
    ERROR_MESSAGE() AS ErrorMessage,
    ERROR_PROCEDURE() AS ErrorProcedure,
    ERROR_LINE() AS ErrorLine;
    end catch
    end
    else
    begin
     select 'Question Does not exist' as ErrorMessag
    end
    end

GO
```


Uses

[dbo].[Choices]

[dbo].[Delete_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--delete course
create proc [dbo].[Delete_Course] @id int
as
if exists(select Course_ID from Course where Course_ID=@id)
begin
    delete from Choices where Question_ID in (select Question_ID from Question where
Course_ID = @id)
    delete from Student_Answer where Question_ID in (select Question_ID from Question
where Course_ID=@id)
    delete from Question where Course_ID =@id
    delete from Instructor_courses where Course_ID =@id
    delete from Student_course where Course_ID=@id
    delete from Exam where Course_ID =@id
    delete from Topic where Course_ID=@id
    delete from Course where Course_ID=@id
end
else select 'Course Not Found'



GO
```

Uses

[dbo].[Choices]
[dbo].[Course]
[dbo].[Exam]
[dbo].[Instructor_Courses]

Project > DESKTOP-7IK5157\SQLEXPRESS > User databases > ExaminationSystem > Programmability > Stored Procedures > dbo.Delete_Course

[dbo].[Question]

[dbo].[Student_Answer]

[dbo].[Student_Course]

[dbo].[Topic]

[dbo].[Delete_Department]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE PROCEDURE [dbo].[Delete_Department]
    @id INT
AS
BEGIN
    IF EXISTS (SELECT dept_id FROM Department WHERE dept_id = @id)
    BEGIN
        IF EXISTS (SELECT Student.Student_ID FROM Student WHERE Student.Dept_ID = @id)
            SELECT 'Cannot delete department: it has associated students' ;
        ELSE IF EXISTS (SELECT Instructor.Instructor_ID FROM Instructor WHERE
Instructor.Dept_ID = @id)
            SELECT 'Cannot delete department: it has associated instructors' ;
        ELSE
            DELETE FROM Department WHERE dept_id = @id;
    END
    ELSE
        SELECT 'Department not found' ;
END;
GO
```

Uses

[dbo].[Department]
[dbo].[Instructor]
[dbo].[Student]

[dbo].[Delete_Exam]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--Delete Exam
CREATE   PROCEDURE [dbo].[Delete_Exam] @ExamID INT
AS
BEGIN
    BEGIN TRY
        DELETE FROM Exam_Question
        WHERE Exam_ID = @ExamID;

        DELETE FROM Exam
        WHERE Exam_ID = @ExamID;

        PRINT 'Exam deleted successfully from the database';
    END TRY
    BEGIN CATCH
        PRINT 'Error while deleting the exam ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

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

[dbo].[Delete_Exam_Question]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@qid	int	4
@ex_id	int	4

SQL Script

```
CREATE PROC [dbo].[Delete_Exam_Question] @qid INT,@ex_id int
    AS
    BEGIN
    set nocount on;
    --checking if the question exists .
    if exists (select c.Question_ID  from Exam_Question c where c.Question_ID =@qid AND
c.Exam_ID=@ex_id)
    begin
    begin try
    delete from dbo.Exam_Question
    where dbo.Exam_Question.Question_ID=@qid
    AND Exam_ID=@ex_id

    end try
    begin catch
    select ERROR_NUMBER() AS ErrorNumber,
    ERROR_MESSAGE() AS ErrorMessage,
    ERROR_PROCEDURE() AS ErrorProcedure,
    ERROR_LINE() AS ErrorLine;
    end catch
    end
    else
    begin
    select'Question Does not exist' as ErrorMessage
    end
END;
GO
```


Uses

[dbo].[Exam_Question]

[dbo].[Delete_Instructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--delete ins by id
create proc [dbo].[Delete_Instructor] @id int
as
if exists(select Instructor_ID from Instructor where Instructor_ID = @id)
    begin
    delete from Instructor where Instructor_ID = @id
    delete from Instructor_Courses where Instructor_ID = @id
    end
else select 'This ID does not exist'
GO
```

Uses

[dbo].[Instructor]
[dbo].[Instructor_Courses]

[dbo].[Delete_Instructor_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ins_id	int	4
@crs_id	int	4

SQL Script

```
--delete instructor_course
CREATE PROCEDURE [dbo].[Delete_Instructor_Course]
    @ins_id INT = 0,
    @crs_id INT = 0
AS
BEGIN
    IF (@ins_id != 0 AND @crs_id = 0)
        DELETE FROM Instructor_Courses WHERE Instructor_ID = @ins_id;
    ELSE IF (@ins_id = 0 AND @crs_id != 0)
        DELETE FROM Instructor_Courses WHERE Course_ID = @crs_id;
    ELSE IF (@ins_id != 0 AND @crs_id != 0)
        DELETE FROM Instructor_Courses WHERE Instructor_ID = @ins_id AND Course_ID =
@crs_id;
END
GO
```

Uses

[dbo].[Instructor_Courses]

[dbo].[Delete_Question]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--4- Delete SP
create proc [dbo].[Delete_Question] @qid int
as
begin
set nocount on ;
if exists(select q.Question_ID  from Question q where q.Question_ID=@qid)
begin
begin try

--deleting the answers of this question first------------
delete from Choices
where Choices.Question_ID=@qid
-- Deleting from exam-question table
DELETE FROM Exam_Question
WHERE Question_ID = @qid;
--deleting the questio
delete from Question
where Question.Question_ID=@qid
--to reset the identity of question id -------------

end try
begin catch
select ERROR_NUMBER() AS ErrorNumber,
ERROR_MESSAGE() AS ErrorMessage,
ERROR_PROCEDURE() AS ErrorProcedure,
ERROR_LINE() AS ErrorLine;
end catch
```



```
    end
    else
    begin
     select 'Question Does not exist' as ErrorMessag
    end
    END


GO
```

Uses

[dbo].[Choices]
[dbo].[Exam_Question]
[dbo].[Question]

[dbo].[Delete_Student]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE PROCEDURE [dbo].[Delete_Student]
    @StudID INT
AS
BEGIN
    IF EXISTS (SELECT Student_ID FROM Student WHERE Student_ID = @StudID)
        BEGIN
            DELETE FROM Student WHERE Student_ID = @StudID
        END
    ELSE SELECT 'No Student Found!';
END
GO
```

Uses

[dbo].[Student]

[dbo].[Delete_Student_Answer]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
-------------------------------------DELETE-------------------------------------
CREATE PROCEDURE [dbo].[Delete_Student_Answer]
    @ExamID INT,
    @QuID INT,
    @StudID INT
AS
BEGIN
    IF EXISTS (SELECT Student_Answer FROM Studend_Answer WHERE Exam_ID = @ExamID AND
Question_ID = @QuID AND Student_ID = @StudID)
    BEGIN
        BEGIN TRY
            DELETE FROM Student_Answer
            WHERE Exam_ID = @ExamID
            AND Question_ID = @QuID
            AND Student_ID = @StudID
        END TRY
        BEGIN CATCH
            SELECT 'There Is An Error Happened!'
        END CATCH
    END
    ELSE SELECT 'There is no Student Answer Matched!'
END
GO
```


Uses

[dbo].[Student_Answer]

[dbo].[Delete_Student_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create    procedure [dbo].[Delete_Student_Course] @StudentID int=-1,@CourseID  int=-1
as
if @CourseID!=-1 and @StudentID!=-1
delete from Student_Course
where Course_ID=@CourseID and Student_ID=@StudentID
GO
```

Uses

[dbo].[Student_Course]

[dbo].[Delete_Topic]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Course_ID	int	4
@TopicName	varchar(50)	50

SQL Script

```
--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-- Delete the topic name
CREATE PROCEDURE [dbo].[Delete_Topic]
    @Course_ID INT,
    @TopicName VARCHAR(50)
AS
BEGIN
    BEGIN TRY
        DELETE FROM Topic
        WHERE Course_ID = @Course_ID
          AND Topic_Name = @TopicName;

        PRINT 'Topic deleted successfully.';
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred while deleting topic.';
    END CATCH
END;

GO
```

Uses

[dbo].[Topic]

[dbo].[ExamAnswers]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@studID	int	4
@ExamID	int	4
@Ans1	varchar(200)	200
@Ans2	varchar(200)	200
@Ans3	varchar(200)	200
@Ans4	varchar(200)	200
@Ans5	varchar(200)	200
@Ans6	varchar(200)	200
@Ans7	varchar(200)	200
@Ans8	varchar(200)	200
@Ans9	varchar(200)	200
@Ans10	varchar(200)	200

SQL Script

```
CREATE PROCEDURE [dbo].[ExamAnswers]
  @studID int,
  @ExamID int,

  @Ans1 varchar(200),
  @Ans2 varchar(200),
  @Ans3 varchar(200),
  @Ans4 varchar(200),
  @Ans5 varchar(200),
  @Ans6 varchar(200),
  @Ans7 varchar(200),
  @Ans8 varchar(200),
  @Ans9 varchar(200),
```



```
  @Ans10 varchar(200)
AS
BEGIN
  DECLARE @examQuestion table (questionID int, RowNum int);


  if Exists(select * from Student_Answer where Exam_ID=@ExamID)
  begin
  select 'Sorry! You already submitted once!'
  return
  end


  INSERT INTO @examQuestion
  SELECT Question_ID, ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) as RowNum
  FROM Exam_Question
  WHERE Exam_ID = @ExamID;

  DECLARE @currentRowNum int = 1;

  WHILE @currentRowNum <= 10
  BEGIN
    DECLARE @currentQuestionID int = (SELECT questionID FROM @examQuestion WHERE RowNum
= @currentRowNum);

    INSERT INTO Student_Answer (Exam_ID, Question_ID, Student_ID, Student_Answer)
    VALUES (@ExamID, @currentQuestionID, @studID,
            CASE
              WHEN @currentRowNum = 1 THEN @Ans1
              WHEN @currentRowNum = 2 THEN @Ans2
              WHEN @currentRowNum = 3 THEN @Ans3
              WHEN @currentRowNum = 4 THEN @Ans4
              WHEN @currentRowNum = 5 THEN @Ans5
              WHEN @currentRowNum = 6 THEN @Ans6
              WHEN @currentRowNum = 7 THEN @Ans7
              WHEN @currentRowNum = 8 THEN @Ans8
              WHEN @currentRowNum = 9 THEN @Ans9
              WHEN @currentRowNum = 10 THEN @Ans10
            END);

    SET @currentRowNum = @currentRowNum + 1;
  END


END
GO
```

Uses

[dbo].[Exam_Question]
[dbo].[Student_Answer]

[dbo].[ExamCorrection]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE PROCEDURE [dbo].[ExamCorrection]
    @StudID INT,
    @ExamID INT
AS
BEGIN
    --Declare Grade
    DECLARE @StdGrade INT;
    SET @StdGrade = 0;

    --Declare Cursor on Studnet Answer Table,
    --Limited to ExamID and StudentID passed to the parameters.
    DECLARE answerCur CURSOR
    FOR
        SELECT Student_Answer,Question_ID
        FROM Student_Answer
        WHERE Student_Answer.Exam_ID = @ExamID
        AND Student_Answer.Student_ID = @StudID
    FOR READ ONLY

    DECLARE @QuID INT
    DECLARE @StdAns varchar(200)
    OPEN answerCur
    FETCH answerCur INTO @StdAns,@QuID
    BEGIN
        --Begin While Loop and set the Cursor to Zero
        WHILE @@FETCH_STATUS=0
        BEGIN
            DECLARE @ModelAnswer VARCHAR(200);
```



```
            --Get the right answer from Question Table
            SELECT @ModelAnswer = Question.Model_Answer
            FROM Question
            WHERE Question.Question_ID = @QuID;

            --Compare Right Answer to The Student Answer
            --If they are the same then Grade++
            IF @ModelAnswer = ISNULL(@StdAns,' ')
            BEGIN
                SET @StdGrade = @StdGrade + 1;
            END

            --Increment Cursor
            FETCH answerCur INTO @StdAns,@QuID
        END
    END

    --Close Cursor
    CLOSE answerCur
    DEALLOCATE answerCur

    --Put The Grade of the Student To Exam Table
    UPDATE Exam
    SET Grade = @StdGrade
    WHERE Exam_ID = @ExamID

END
GO
```

Uses

[dbo].[Exam]
[dbo].[Question]
[dbo].[Student_Answer]

[dbo].[ExamGeneration]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE PROCEDURE [dbo].[ExamGeneration]
    @ExamID INT
AS
BEGIN
    DECLARE @CourseID INT;
    DECLARE @TFcount INT;
    DECLARE @MCQcount INT;

    IF EXISTS (SELECT * FROM Exam_Question WHERE Exam_Question.Exam_ID = @ExamID)return

     SELECT @CourseID = Exam_Structure.Course_ID FROM Exam , Exam_Structure WHERE
Exam_ID = @ExamID and Exam.Structure_ID = Exam_Structure.Structure_ID;
     SELECT @TFcount = Exam_Structure.TF_Count FROM Exam , Exam_Structure WHERE Exam_ID
= @ExamID and Exam.Structure_ID = Exam_Structure.Structure_ID;
     SELECT @MCQcount = Exam_Structure.MCQ_Count from  Exam , Exam_Structure WHERE
Exam_ID = @ExamID and Exam.Structure_ID = Exam_Structure.Structure_ID;

    IF (@CourseID IS NULL)
    BEGIN
        SELECT 'Invalid Exam ID';
        RETURN;
    END;




    DECLARE @ExamQuestion TABLE (QuestionID INT);

    -- Insert T/F Questions
    INSERT INTO @ExamQuestion (QuestionID)
```



```
    SELECT TOP (@TFcount) Question_ID
    FROM Question
    WHERE Type = 'TF' AND Course_ID = @CourseID
    ORDER BY NEWID();

    -- Insert MCQ Questions
    INSERT INTO @ExamQuestion (QuestionID)
    SELECT TOP (@MCQcount) Question_ID
    FROM Question
    WHERE Type = 'MCQ' AND Course_ID = @CourseID
    ORDER BY NEWID();

    -- Insert generated questions into Exam_Question table for the provided Exam_ID
    INSERT INTO Exam_Question (Exam_ID, Question_ID)
    SELECT @ExamID, QuestionID FROM @ExamQuestion;
END
GO
```

Uses

[dbo].[Exam]
[dbo].[Exam_Question]
[dbo].[Exam_Structure]
[dbo].[Question]

[dbo].[Get_Exam_With_Choices_Report]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE PROCEDURE [dbo].[Get_Exam_With_Choices_Report]
    @ExamID INT
AS
BEGIN
    DECLARE crs CURSOR FOR
        SELECT Question_ID
        FROM Exam_Question
        WHERE Exam_ID = @ExamID
        FOR READ ONLY

    DECLARE @QuID INT
    OPEN crs
    FETCH crs INTO @QuID

    BEGIN
        WHILE @@FETCH_STATUS = 0
        BEGIN
            -- Select Question row
            SELECT Question
            FROM Question
            WHERE Question_ID = @QuID
            union All
            -- Select choices for the current question
            SELECT Choices
            FROM Choices
            WHERE Question_ID = @QuID
            FETCH crs INTO @QuID
        END
    END
```



```
    CLOSE crs
    DEALLOCATE crs
END
GO
```

Uses

[dbo].[Choices]
[dbo].[Exam_Question]
[dbo].[Question]

[dbo].[GetExamDetails]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   PROCEDURE [dbo].[GetExamDetails]
    @Exam_ID INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @ExamDate DATE;
    DECLARE @ExamTime TIME;
    DECLARE @ExamDuration INT;

    -- Retrieve exam date, time, and duration based on the provided Exam_ID
    SELECT @ExamDate = es.Date,
           @ExamTime = es.Time,
           @ExamDuration = es.Duration
    FROM Exam e
    INNER JOIN Exam_Structure es ON e.Structure_ID = es.Structure_ID
    WHERE e.Exam_ID = @Exam_ID;

    -- Return the exam date, time, and duration
    SELECT @ExamDate AS ExamDate,
           @ExamTime AS ExamTime,
           @ExamDuration AS ExamDuration;
END;
GO
```

Uses

[dbo].[Exam]

[dbo].[Exam_Structure]

[dbo].[GetInstCrsStuds_Report]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
Create PROCEDURE [dbo].[GetInstCrsStuds_Report]
    @InstID INT
AS
BEGIN
    SELECT
        Course.Crs_Name,
        COUNT(Student_Course.Student_ID) as Num_Students
    FROM
        Course
        LEFT JOIN
        Student_Course ON Course.Course_ID = Student_Course.Course_ID
        LEFT JOIN
        Instructor_Courses ON Instructor_Courses.Course_ID = Course.Course_ID
    WHERE
        Instructor_Courses.Instructor_ID = @InstID
    GROUP BY
        Course.Crs_Name;
END
GO
```

Uses

[dbo].[Course]
[dbo].[Instructor_Courses]
[dbo].[Student_Course]

[dbo].[GetQuestionsWithStudentAnswersReport]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create proc [dbo].[GetQuestionsWithStudentAnswersReport] @QuID int, @ExamID int
as
select Question.Question as TheQuestions , Student_Answer.Student_Answer as Student-
Answers
from Exam_Question , Student_Answer ,Question
where Question.Question_ID = Exam_Question.Question_ID
and Student_Answer.Exam_ID=Exam_Question.Exam_ID
and Question.Question_ID=@QuID
and Exam_Question.Exam_ID=@ExamID
GO
```

Uses

[dbo].[Exam_Question]
[dbo].[Question]
[dbo].[Student_Answer]

[dbo].[GetStudentGradesReport]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE PROCEDURE [dbo].[GetStudentGradesReport] @StudentID INT
AS
BEGIN
    SELECT Course.Crs_Name,Exam.Grade
    FROM Course,Exam_Structure,Exam
    WHERE
        Exam.Grade IS NOT NULL
        AND Exam.Student_ID = @StudentID
        and Exam_Structure.Course_ID=Course.Course_ID
        and exam.Structure_ID=Exam_Structure.Structure_ID
END;

GO
```

Uses

[dbo].[Course]
[dbo].[Exam]
[dbo].[Exam_Structure]

[dbo].[GetStudentInformationReport]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE   proc [dbo].[GetStudentInformationReport] @dept_num int
as
begin
    SET NOCOUNT ON;

    -- Checking if the department exists
    if exists (select d.Dept_ID
               from Department d
               where dept_id = @dept_num )
    begin
        begin try
            select s.Student_ID AS ID ,
            ISNULL(s.First_Name,'NA') AS First_Name,
            ISNULL(s.Second_Name,'NA') AS Last_Name,
            ISNULL(s.Email,'NA') AS Email,
            s.Dept_ID as Department
            from Student s
            where s.Dept_ID=@dept_num

        end try
        begin catch

            SELECT ERROR_NUMBER() AS ErrorNumber,
                   ERROR_MESSAGE() AS ErrorMessage,
                   ERROR_PROCEDURE() AS ErrorProcedure,
                   ERROR_LINE() AS ErrorLine;


        end catch;
```

```
    end
    else
    begin
        print 'Department not found';
    end
end
GO
```

Uses

[dbo].[Department]
[dbo].[Student]

[dbo].[GetTopicNameReport]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE   proc [dbo].[GetTopicNameReport]
    @crsId int
as
begin
    SET NOCOUNT ON;

    if exists (select Course_ID
               from Course
               where Course_ID = @crsId)
    begin
        begin try
            select *
            from Topic
            where Course_ID = @crsId;
        end try
        begin catch
            select ERROR_NUMBER() AS ErrorNumber,
                   ERROR_MESSAGE() AS ErrorMessage,
                   ERROR_PROCEDURE() AS ErrorProcedure,
                   ERROR_LINE() AS ErrorLine;
        end catch;
    end
    else
    begin
        select 'Course not found' AS ErrorMessage;
    end
end;
GO
```


Uses

[dbo].[Course]

[dbo].[Topic]

[dbo].[Insert_Choices]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@QuestionID	int	4
@Choice1	varchar(50)	50
@Choice2	varchar(50)	50
@Choice3	varchar(50)	50
@Choice4	varchar(50)	50

SQL Script

```
    ------------------------------------------------------------------
    --table choices ----

    --1-insert choicese
    CREATE PROCEDURE [dbo].[Insert_Choices] @QuestionID INT ,
    @Choice1 VARCHAR(50),
    @Choice2 VARCHAR(50),
    @Choice3 VARCHAR(50),
    @Choice4 VARCHAR(50)
AS
BEGIN
set nocount on;

    -- Insert into Question table
    INSERT INTO dbo.Choices
    (
        Question_ID,
        Choices
    )
    VALUES
    (@QuestionID, @Choice1),
            (@QuestionID, @Choice2),
            (@QuestionID, @Choice3),
```



```
            (@QuestionID, @Choice4);
            END;



GO
```

Uses

[dbo].[Choices]

[dbo].[Insert_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@crsname	nvarchar(15)	30

SQL Script

```
CREATE PROCEDURE [dbo].[Insert_Course]
    @crsname NVARCHAR(15)
AS
BEGIN
    BEGIN TRY
        INSERT INTO Course(Crs_Name)
        VALUES (@crsname)
    END TRY
    BEGIN CATCH
        -- Error message
        SELECT
            'An Error happened while inserting: ' + ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END
GO
```

Uses

[dbo].[Course]

[dbo].[Insert_Department]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	nvarchar(20)	40

SQL Script

```
create proc [dbo].[Insert_Department] @name nvarchar(20)
as
insert into Department values(@name)
GO
```

Uses

[dbo].[Department]

[dbo].[Insert_Exam_Question]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@qid	int	4
@ex_id	int	4

SQL Script

```
--2- Insert SP
create proc [dbo].[Insert_Exam_Question] @qid int , @ex_id int
    as
    BEGIN
    set nocount on;
    --checking if the question exists .
    if exists (select c.Question_ID  from Exam_Question c where c.Question_ID =@qid)
    begin
    begin try
    -- Insert into Question table
    INSERT INTO Exam_Question(Question_ID,Exam_ID)
    VALUES (@qid,@ex_id);
    end try
    begin catch
    select ERROR_NUMBER() AS ErrorNumber,
    ERROR_MESSAGE() AS ErrorMessage,
    ERROR_PROCEDURE() AS ErrorProcedure,
    ERROR_LINE() AS ErrorLine;
    end catch
    end
    else
    begin
    select'Question Does not exist' as ErrorMessage
    end

END;
```



```
GO
```

Uses

[dbo].[Exam_Question]

[dbo].[Insert_Instructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ins_id	int	4
@fname	nvarchar(10)	20
@lname	nvarchar(10)	20
@email	nvarchar(70)	140
@did	int	4

SQL Script

```
CREATE PROCEDURE [dbo].[Insert_Instructor]
    @ins_id INT,
    @fname NVARCHAR(10),
    @lname NVARCHAR(10),
    @email NVARCHAR(70) = NULL,
    @did INT
AS
BEGIN
    BEGIN TRY
        INSERT INTO Instructor (Instructor_ID, First_Name, Second_Name, Email, Dept_ID)
        VALUES (@ins_id, @fname, @lname, @email, @did);
    END TRY
    BEGIN CATCH
        --Error message
        SELECT
            'An Error happened while inserting: ' + ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END
GO
```


Uses

[dbo].[Instructor]

[dbo].[Insert_Instructor_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ins_id	int	4
@crs_id	int	4

SQL Script

```
CREATE PROCEDURE [dbo].[Insert_Instructor_Course]
    @ins_id INT,
    @crs_id INT
AS
BEGIN
    BEGIN TRY
        INSERT INTO Instructor_Courses (Course_ID, Instructor_ID)
        VALUES (@crs_id, @ins_id);
    END TRY
    BEGIN CATCH
        --Eroor message
        SELECT
            'An Error happened while inserting: ' + ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END
GO
```

Uses

[dbo].[Instructor_Courses]

[dbo].[Insert_Question]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Type	varchar(3)	3
@QuestionText	varchar(200)	200
@ModelAnswer	varchar(200)	200
@CourseID	int	4
@Choice1	varchar(200)	200
@Choice2	varchar(200)	200
@Choice3	varchar(200)	200
@Choice4	varchar(200)	200

SQL Script

```
CREATE PROCEDURE [dbo].[Insert_Question]
    @Type VARCHAR(3),
    @QuestionText VARCHAR(200),
    @ModelAnswer VARCHAR(200),
    @CourseID INT,
    @Choice1 VARCHAR(200),
    @Choice2 VARCHAR(200),
    @Choice3 VARCHAR(200),
    @Choice4 VARCHAR(200)
AS
BEGIN
    -- Insert into Question table
    INSERT INTO Question (Type, Question, Model_Answer, Course_ID)
    VALUES (@Type, @QuestionText, @ModelAnswer, @CourseID);

    -- Get the generated Question_ID
    DECLARE  @QuestionID int  = SCOPE_IDENTITY();

    -- Insert into Choices table if the question type is MCQ
    IF @Type = 'MCQ'
```



```
    BEGIN
        INSERT INTO Choices (Question_ID, Choices)
        VALUES
            (@QuestionID, @Choice1),
            (@QuestionID, @Choice2),
            (@QuestionID, @Choice3),
            (@QuestionID, @Choice4);
    END
END;
GO
```

Uses

[dbo].[Choices]
[dbo].[Question]

[dbo].[Insert_Student]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@FirstName	varchar(15)	15
@SecondName	varchar(15)	15
@Email	varchar(50)	50
@Password	varchar(50)	50
@DeptID	int	4

SQL Script

```
CREATE PROCEDURE [dbo].[Insert_Student]
    @FirstName VARCHAR(15),
    @SecondName VARCHAR(15),
    @Email VARCHAR(50),
    @Password VARCHAR(50),
    @DeptID INT = NULL
AS
BEGIN
    --Check If There Is Any Error Will Happen Using TRY-CATCH
    BEGIN TRY
        INSERT INTO Student
        VALUES (@FirstName, @SecondName, @Email, @Password, @DeptID)
    END TRY

    BEGIN CATCH
        SELECT 'There Is An Error Happened!'
    END CATCH
END
GO
```

Uses

[dbo].[Student]

[dbo].[Insert_Student_Answer]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ExamID	int	4
@QuID	int	4
@StudID	int	4
@StudAns	varchar(200)	200

SQL Script

```
-------------------------------------INSERT----------------------------------------
CREATE PROCEDURE [dbo].[Insert_Student_Answer]
    @ExamID INT,
    @QuID INT,
    @StudID INT,
    @StudAns varchar(200)
AS
BEGIN
    BEGIN TRY
        INSERT INTO Student_Answer
        VALUES (@ExamID, @QuID, @StudID, @StudAns)
    END TRY
    BEGIN CATCH
        SELECT 'There Is An Error Happened!'
    END CATCH
END
GO
```

Uses

[dbo].[Student_Answer]

[dbo].[Insert_Student_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create procedure [dbo].[Insert_Student_Course] @StudentID int,@CourseID int as
begin try
insert into student_course
values (@StudentID,@CourseID)
end try
begin catch
select 'error in insert'
end catch
GO
```

Uses

[dbo].[Student_Course]

[dbo].[Insert_Topic]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@CourseID	int	4
@CourseName	varchar(20)	20

SQL Script

```
CREATE   PROCEDURE [dbo].[Insert_Topic]  @CourseID int, @CourseName VARCHAR(20)
AS
BEGIN
    BEGIN TRY
        INSERT INTO Topic (Course_ID, Topic_Name)
        VALUES (@CourseID, @CourseName)
    END TRY
    BEGIN CATCH
        PRINT 'Invalid data';
    END CATCH
END;
GO
```

Uses

[dbo].[Topic]

[dbo].[Select_All_Course_With_Topics]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
CREATE   PROCEDURE [dbo].[Select_All_Course_With_Topics]
AS
BEGIN
    BEGIN TRY
        SELECT Course.Crs_Name AS CourseName, Topic.Topic_Name AS Topics
        FROM Course , Topic
        WHERE Course.Course_ID = Topic.Course_ID;
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred while selecting courses with topics.';
    END CATCH
END;
GO
```

Uses

[dbo].[Course]
[dbo].[Topic]

[dbo].[Select_Choices]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
    --3-select choices -------
    create proc [dbo].[Select_Choices] @qid int
    as
    begin
    set nocount on;
    --checking if the question exists .
    if exists (select c.Question_ID  from Choices c where c.Question_ID =@qid)
    begin
    begin try
    select c.Choices
    from Choices c
    where c.Question_ID=@qid
    end try
    begin catch
    select ERROR_NUMBER() AS ErrorNumber,
    ERROR_MESSAGE() AS ErrorMessage,
    ERROR_PROCEDURE() AS ErrorProcedure,
    ERROR_LINE() AS ErrorLine;
    end catch
    end
else
    begin
    select'Question Does not exist' as ErrorMessage
    end
end;

GO
```


Uses

[dbo].[Choices]

[dbo].[Select_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE proc [dbo].[Select_Course] @crs_id int =-1
as
    if(@crs_id!=-1)
        select Course_ID , Crs_Name as CourseName from Course where Course_ID=@crs_id
    else
        select Course_ID , Crs_Name as CourseName from Course
GO
```

Uses

[dbo].[Course]

[dbo].[Select_Course_With_Topics]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@CourseName	varchar(20)	20

SQL Script

```
CREATE   PROCEDURE [dbo].[Select_Course_With_Topics]
    @CourseName VARCHAR(20)
AS
BEGIN
    BEGIN TRY
        SELECT Course.Crs_Name AS CourseName, Topic.Topic_Name AS Topics
        FROM Course
        INNER JOIN Topic ON Course.Course_ID = Topic.Course_ID
        WHERE Course.Crs_Name = @CourseName;
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred while selecting courses with topics.';
    END CATCH
END;
GO
```

Uses

[dbo].[Course]
[dbo].[Topic]

[dbo].[Select_Department]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create proc [dbo].[Select_Department] @id int =-1 , @name nvarchar(50)=' '
as
if @id=-1 and @name!=' '
begin
if exists (select dept_name from Department where dept_name=@name )
select dept_id as [dept id] , dept_name as [dept name] from Department where
dept_name=@name
else select 'department not found'
end
else if @id!=-1 and @name=' '
begin
if exists (select dept_id from Department where dept_id=@id )
select dept_id as [dept id] , dept_name as [dept name] from Department where
dept_id=@id
else select 'department not found'
end
else
select dept_id as [dept id] , dept_name as [dept name] from Department
GO
```

Uses

[dbo].[Department]

[dbo].[Select_Exam_Question]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
    ------------------------------------------------------------------------------
-
    --table Exam-Question
    --1 Select SP
    CREATE   proc [dbo].[Select_Exam_Question] @Exam_id int
    as
    select q.* from   Question q ,Exam_Question e

    where
    e.Exam_ID= @Exam_id and q.Question_ID =e.Question_ID
GO
```

Uses

[dbo].[Exam_Question]
[dbo].[Question]

[dbo].[Select_Exams_By_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
-- Select Exams for a Specific Course
CREATE   PROCEDURE [dbo].[Select_Exams_By_Course]
    @Course_ID INT
AS
BEGIN
    BEGIN TRY
        SELECT *
        FROM Exam ,Exam_Structure
        WHERE Exam_Structure.Course_ID = @Course_ID and Exam.Structure_ID = Exam_-
Structure.Structure_ID;
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred while selecting exams by course.';
    END CATCH
END;
GO
```

Uses

[dbo].[Exam]
[dbo].[Exam_Structure]

[dbo].[Select_InstructorCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ins_id	int	4
@crs_id	int	4

SQL Script

```
--select from InstructorCourse
CREATE PROCEDURE [dbo].[Select_InstructorCourse]
    @ins_id INT = 0,
    @crs_id INT = 0
AS
BEGIN
    IF (@ins_id = 0 AND @crs_id != 0)
    BEGIN
        -- Query for filtering by course ID only
        SELECT
            C.Course_ID,
            C.Crs_Name AS CourseName,
            I.Instructor_ID,
            I.First_Name + ' ' + I.Second_Name AS Full_Name
        FROM
            Instructor_Courses IC
            INNER JOIN Course C ON IC.Course_ID = C.Course_ID
            INNER JOIN Instructor I ON IC.Instructor_ID = I.Instructor_ID
        WHERE
            IC.Course_ID = @crs_id;
    END
    ELSE IF (@crs_id = 0 AND @ins_id != 0)
    BEGIN
        -- Query for filtering by instructor ID only
        SELECT
            C.Course_ID,
            C.Crs_Name AS CourseName,
            I.Instructor_ID,
```



```
            I.First_Name + ' ' + I.Second_Name AS Full_Name
        FROM
            Instructor_Courses IC
            INNER JOIN Course C ON IC.Course_ID = C.Course_ID
            INNER JOIN Instructor I ON IC.Instructor_ID = I.Instructor_ID
        WHERE
            I.Instructor_ID = @ins_id;
    END
    ELSE IF (@ins_id = 0 AND @crs_id = 0)
    BEGIN
        -- Query without any filtering
        SELECT
            C.Course_ID,
            C.Crs_Name AS CourseName,
            I.Instructor_ID,
            I.First_Name + ' ' + I.Second_Name AS Full_Name
        FROM
            Instructor_Courses IC
            INNER JOIN Course C ON IC.Course_ID = C.Course_ID
            INNER JOIN Instructor I ON IC.Instructor_ID = I.Instructor_ID;
    END
    ELSE
    BEGIN
        -- Query for filtering by both course ID and instructor ID
        SELECT
            C.Course_ID,
            C.Crs_Name AS CourseName,
            I.Instructor_ID,
            I.First_Name + ' ' + I.Second_Name AS Full_Name
        FROM
            Instructor_Courses IC
            INNER JOIN Course C ON IC.Course_ID = C.Course_ID
            INNER JOIN Instructor I ON IC.Instructor_ID = I.Instructor_ID
        WHERE
            IC.Course_ID = @crs_id
            AND I.Instructor_ID = @ins_id;
    END
END
GO
```

Uses

[dbo].[Course]
[dbo].[Instructor]
[dbo].[Instructor_Courses]

[dbo].[Select_InstructorInfo]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Ins_ID	int	4
@Ins_name	varchar(20)	20

SQL Script

```
CREATE Procedure [dbo].[Select_InstructorInfo] @Ins_ID int=-1 , @Ins_name varchar(20)='
'
as
    if @Ins_ID = -1 and @Ins_name !=' '
        begin
        if exists(select First_Name from Instructor where First_Name = @Ins_name)
            select I.Instructor_ID, I.First_Name +' '+I.Second_Name as Full_Name
            ,I.Email,D.Dept_Name Department,I.Dept_ID Department_ID
            from Instructor I,Department D
            where D.Dept_ID = I.Dept_ID
            and I.First_Name=@Ins_name
        else select 'No Matching Instructor'
        end
    else if @Ins_ID != -1 and @Ins_name =' '
        begin
        if exists(select Instructor_ID from Instructor where Instructor_ID = @Ins_ID)
            select I.Instructor_ID, I.First_Name +' '+I.Second_Name as Full_Name
            ,I.Email,D.Dept_Name Department,I.Dept_ID Department_ID
            from Instructor I,Department D
            where D.Dept_ID = I.Dept_ID
            and I.Instructor_ID=@Ins_ID
        else select 'No Matching instructor'
        end
    else
            select I.Instructor_ID, I.First_Name +' '+I.Second_Name as Full_Name
            ,I.Email,D.Dept_Name Department,I.Dept_ID Department_ID
            from Instructor I,Department D
            where D.Dept_ID = I.Dept_ID
```



```
GO
```

Uses

[dbo].[Department]

[dbo].[Instructor]

[dbo].[Select_Question]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--Question table
--1- Select SP
create proc [dbo].[Select_Question] @q_id int
as
begin
set nocount on;
--checking if the question exists .
if exists (select q.Question_ID  from Question q where q.Question_ID =@q_id)
    begin
    begin try
    select q. Question_ID as ID,
    isnull(q.Type,'NA') as Type,
    isnull(q.Question,'NA') as Question,
    isnull(c.Crs_Name,'NA') as Course_Name,
    isnull(q.Model_Answer,'NA') as Model_Answer
    from Question q , Course c
    where  c.Course_ID =q.Course_ID
    and q.Question_ID=@q_id
    end try
    begin catch
    select ERROR_NUMBER() AS ErrorNumber,
    ERROR_MESSAGE() AS ErrorMessage,
    ERROR_PROCEDURE() AS ErrorProcedure,
    ERROR_LINE() AS ErrorLine;
    end catch
    end
else
    begin
    select'Question Does not exist' as ErrorMessage
```



```
    end
end;

GO
```

Uses

[dbo].[Course]
[dbo].[Question]

[dbo].[Select_Student]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StudID	int	4
@StudFName	varchar(15)	15

SQL Script

```
CREATE PROCEDURE [dbo].[Select_Student]
    @StudID int =-1,
    @StudFName varchar(15) = ' '
AS
BEGIN
    --If The User only Entered The Student ID
    IF(@StudID != -1 AND @StudFName = ' ')
    BEGIN
        --Check First If He Exists
        IF EXISTS (SELECT First_Name FROM Student WHERE Student_ID = @StudID)
            SELECT Student_ID, First_Name + ' ' + Second_Name as [Full Name], Email,
                ISNULL(Department.Dept_Name, 'NA') as DeptName
            FROM Student LEFT JOIN Department
            ON Student.Dept_ID = Department.Dept_ID
            WHERE Student_ID = @StudID

        ELSE SELECT 'No Student Found!';
    END

    --If The User only Entered The Student First Name
    ELSE IF(@StudID = -1 AND @StudFName != ' ')
    BEGIN
        --Check First If He Exists
        IF EXISTS (SELECT First_Name FROM Student WHERE First_Name = @StudFName)
            SELECT Student_ID, First_Name + ' ' + Second_Name as [Full Name], Email,
                ISNULL(Department.Dept_Name, 'NA') as DeptName
            FROM Student LEFT JOIN Department
            ON Department.Dept_Name = Student.Dept_ID
```



```
            WHERE First_Name = @StudFName

        ELSE SELECT 'No Student Found!';
    END

    --If The User Didn't Enter Any Info, Then Print All
    ELSE
    BEGIN
        SELECT Student_ID, First_Name + ' ' + Second_Name as [Full Name], Email,
            ISNULL(Department.Dept_Name, 'NA') as DeptName
        FROM Student LEFT JOIN Department
        ON Student.Dept_ID = Department.Dept_ID
    END
END
GO
```

Uses

[dbo].[Department]
[dbo].[Student]

[dbo].[Select_Student_Answer]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--Table Student_Answer


---------------------------------------SELECT-----------------------------------

CREATE PROCEDURE [dbo].[Select_Student_Answer]
    @StudID INT = -1,
    @ExamID INT = -1,
    @QuID INT = -1
AS
BEGIN
    BEGIN TRY
        IF(@StudID = -1 AND @ExamID != -1 AND @QuID != -1)
        BEGIN
            SELECT *
            FROM Student_Answer
            WHERE Exam_ID = @ExamID
            AND Question_ID = @QuID
        END

        ELSE IF(@StudID = -1 AND @ExamID = -1 AND @QuID != -1)
        BEGIN
            SELECT *
            FROM Student_Answer
            WHERE Question_ID = @QuID
        END
```



```
        ELSE IF(@StudID = -1 AND @ExamID != -1 AND @QuID = -1)
        BEGIN
            SELECT *
            FROM Student_Answer
            WHERE Exam_ID = @ExamID
        END

        ELSE IF(@StudID = -1 AND @ExamID = -1 AND @QuID = -1)
        BEGIN
            SELECT *
            FROM Student_Answer
        END

        ELSE IF(@StudID != -1 AND @ExamID = -1 AND @QuID = -1)
        BEGIN
            SELECT *
            FROM Student_Answer
            WHERE Student_ID = @StudID
        END

        ELSE IF(@StudID != -1 AND @ExamID != -1 AND @QuID = -1)
        BEGIN
            SELECT *
            FROM Student_Answer
            WHERE Student_ID = @StudID
            AND Exam_ID = @ExamID
        END

        ELSE IF(@StudID != -1 AND @ExamID = -1 AND @QuID != -1)
        BEGIN
            SELECT *
            FROM Student_Answer
            WHERE Student_ID = @StudID
            AND Question_ID = @QuID
        END

        ELSE IF(@StudID != -1 AND @ExamID != -1 AND @QuID != -1)
        BEGIN
            SELECT *
            FROM Student_Answer
            WHERE Student_ID = @StudID
            AND Question_ID = @QuID
            AND Exam_ID = @ExamID
        END
    END TRY
    BEGIN CATCH
        SELECT 'There is an Error Heppened'
    END CATCH
END
GO
```


Uses

[dbo].[Student_Answer]

[dbo].[Select_Student_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   procedure [dbo].[Select_Student_Course] @StudentID  int =-1,@CourseID   int =-
1  as
if (@StudentID !=-1 and @CourseID  !=-1)
select Student_Course.Student_ID ,Student_Course.Course_ID , Student.First_-
Name+Student.Second_Name as StudentName ,Course.Crs_Name as CourseName
from Student,Course,Student_Course
where Student_Course.Student_ID = Student.Student_ID and Student_Course.Course_-
ID=Course.Course_ID and Student_Course.Student_ID=@StudentID and Student_-
Course.Course_ID =@CourseID


else if  @StudentID!=-1
select Student_Course.Student_ID ,Student_Course.Course_ID , Student.First_-
Name+Student.Second_Name as StudentName ,Course.Crs_Name as CourseName
from Student,Course,Student_Course
where Student_Course.Student_ID = Student.Student_ID and Student_Course.Course_-
ID=Course.Course_ID and Student_Course.Student_ID=@StudentID


else if @CourseID !=-1
select Student_Course.Student_ID ,Student_Course.Course_ID , Student.First_-
Name+Student.Second_Name as StudentName ,Course.Crs_Name as CourseName
from Student,Course,Student_Course
where Student_Course.Student_ID = Student.Student_ID and Student_Course.Course_-
ID=Course.Course_ID  and Student_Course.Course_ID =@CourseID
GO
```

Uses

[dbo].[Course]

Project > DESKTOP-7IK5157\SQLEXPRESS > User databases > ExaminationSystem > Programmability > Stored Procedures > dbo.Select_Student_Course

[dbo].[Student]

[dbo].[Student_Course]

[dbo].[Select_Topic]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE   PROCEDURE [dbo].[Select_Topic]
    @Course_ID INT = -1
AS
BEGIN
    BEGIN TRY
    IF (@Course_ID != -1)
        SELECT Topic_Name
        FROM Topic
        WHERE Course_ID = @Course_ID;
    ELSE
    SELECT Course.Crs_Name , Topic_Name
    FROM dbo.Topic,dbo.Course
    WHERE topic.Course_ID = course.Course_ID
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred while selecting topic name by ID.';
    END CATCH
END;
GO
```

Uses

[dbo].[Course]
[dbo].[Topic]

[dbo].[Select_TopicNameById]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE   PROCEDURE [dbo].[Select_TopicNameById]
    @Course_ID INT
AS
BEGIN
    BEGIN TRY


        SELECT Topic_Name
        FROM Topic
        WHERE Course_ID = @Course_ID;
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred while selecting topic name by ID.';
    END CATCH
END;
GO
```

Uses

[dbo].[Topic]

[dbo].[Select_TopicNames]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
CREATE   PROCEDURE [dbo].[Select_TopicNames]
AS
BEGIN
    BEGIN TRY
        SELECT Topic_Name
        FROM Topic;
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred while selecting all topic names.';
    END CATCH
END;
GO
```

Uses

[dbo].[Topic]

[dbo].[SelectStudentExams]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
 create     proc  [dbo].[SelectStudentExams]
as
begin
select Exam.Exam_ID, Course.Crs_Name ,Exam_Structure.Date,Exam_Structure.time,Exam_-
Structure.Duration ,Exam.Grade
from  Course  , Exam , Exam_Structure
where
 Course.Course_ID=Exam_Structure.Course_ID and exam.Structure_ID=Exam_-
Structure.Structure_ID


 end
GO
```

Uses

[dbo].[Course]
[dbo].[Exam]
[dbo].[Exam_Structure]

[dbo].[Update_Choices]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@qid	int	4
@oldChoice1	varchar(50)	50
@oldChoice2	varchar(50)	50
@oldChoice3	varchar(50)	50
@oldChoice4	varchar(50)	50
@Choice1	varchar(50)	50
@Choice2	varchar(50)	50
@Choice3	varchar(50)	50
@Choice4	varchar(50)	50

SQL Script

```
-----------------------------------------------------------------------
--2-update choices
create proc [dbo].[Update_Choices]
@qid int ,
 @oldChoice1 VARCHAR(50),
@oldChoice2 VARCHAR(50),
@oldChoice3 VARCHAR(50),
@oldChoice4 VARCHAR(50),
@Choice1 VARCHAR(50),
@Choice2 VARCHAR(50),
@Choice3 VARCHAR(50),
@Choice4 VARCHAR(50)
as
begin
set nocount on;
if exists(select c.Question_ID  from Choices c where c.Question_ID=@qid)
begin
begin try
update Choices
```



```
    set Choices.Choices=@Choice1
    where
    Choices.Question_ID=@qid
    and Choices.Choices=@oldChoice1
    update Choices
    set Choices.Choices=@Choice2
    where
    Choices.Question_ID=@qid
    and Choices.Choices=@oldChoice2
    update Choices
    set Choices.Choices=@Choice3
    where
    Choices.Question_ID=@qid
    and Choices.Choices=@oldChoice3
    update Choices
    set Choices.Choices=@Choice4
    where
    Choices.Question_ID=@qid
    and Choices.Choices=@oldChoice4
    end try
    begin catch
    select ERROR_NUMBER() AS ErrorNumber,
    ERROR_MESSAGE() AS ErrorMessage,
    ERROR_PROCEDURE() AS ErrorProcedure,
    ERROR_LINE() AS ErrorLine;
    end catch
    end
    else
    begin
    select 'Question Does not exist' as ErrorMessag
    end
    END

GO
```

Uses

[dbo].[Choices]

[dbo].[Update_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE PROCEDURE [dbo].[Update_Course]
    @id INT,
    @name NVARCHAR(50)
AS
BEGIN
    IF EXISTS (SELECT Course_ID FROM Course WHERE Course_ID = @id)
    BEGIN
        UPDATE Course SET Crs_Name = @name WHERE Course_ID = @id;
        SELECT 'Course updated successfully' AS Status;
    END
    ELSE
        SELECT 'Course not found' AS Status;
END
GO
```

Uses

[dbo].[Course]

[dbo].[Update_Department]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@DeptID	int	4
@NewDeptName	varchar(200)	200

SQL Script

```
CREATE PROCEDURE [dbo].[Update_Department]
    @DeptID INT,
    @NewDeptName VARCHAR(200)
AS
BEGIN
    BEGIN TRY
        UPDATE Department
        SET Dept_Name = @NewDeptName
        WHERE Dept_ID = @DeptID;

        IF @@ROWCOUNT = 0
        BEGIN
            SELECT 'Department not found' ;
        END
    END TRY
    BEGIN CATCH
        SELECT 'An error occurred' ;
    END CATCH;
END;
GO
```

Uses

[dbo].[Department]

[dbo].[Update_Exam_Grade]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Exam_ID	int	4
@NewGrade	int	4

SQL Script

```
-- Update Exam Grade
CREATE   PROCEDURE [dbo].[Update_Exam_Grade]
    @Exam_ID INT,
    @NewGrade INT
AS
BEGIN
    BEGIN TRY
        UPDATE Exam
        SET Grade = @NewGrade
        WHERE Exam_ID = @Exam_ID;

        PRINT 'Exam grade updated successfully.';
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred while updating exam grade.';
    END CATCH
END;
GO
```

Uses

[dbo].[Exam]

[dbo].[Update_Exam_Question]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@old_qid	int	4
@old_ex_id	int	4
@new_qid	int	4

SQL Script

```
CREATE proc [dbo].[Update_Exam_Question] @old_qid int , @old_ex_id int,@new_qid int
    as
    begin
    set nocount on;
    if exists(select c.Question_ID  from Exam_Question c where c.Question_ID=@old_qid
and c.Exam_ID=@old_ex_id)
    begin
    begin try

    update Exam_Question
    set Question_ID=@new_qid
    where
    Exam_Question.Question_ID=@old_qid
    AND
    dbo.Exam_Question.Exam_ID=@old_ex_id

    end try
    begin catch
    select ERROR_NUMBER() AS ErrorNumber,
    ERROR_MESSAGE() AS ErrorMessage,
    ERROR_PROCEDURE() AS ErrorProcedure,
    ERROR_LINE() AS ErrorLine;
    end catch
    end
    else
    begin
    select 'Question Does not exist' as ErrorMessag
```



```
    end
    END;
GO
```

Uses

[dbo].[Exam_Question]

[dbo].[Update_Instructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ins_ID	int	4
@fname	nvarchar(50)	100
@email	varchar(50)	50
@d_id	int	4

SQL Script

```
CREATE PROCEDURE [dbo].[Update_Instructor]
    @ins_ID INT,
    @fname NVARCHAR(50) = '',
    @email VARCHAR(50) = '',
    @d_id INT = 0
AS
BEGIN
    IF EXISTS (SELECT Instructor_ID FROM Instructor WHERE Instructor_ID = @ins_ID)
    BEGIN
        BEGIN TRY
            IF (@fname != '')
                UPDATE Instructor SET First_Name = @fname WHERE Instructor_ID = @ins_-
ID;
            IF (@email != '')
                UPDATE Instructor SET Email = @email WHERE Instructor_ID = @ins_ID;
            IF (@d_id != 0)
                UPDATE Instructor SET Dept_ID = @d_id WHERE Instructor_ID = @ins_ID;
        END TRY
        BEGIN CATCH
            SELECT 'Invalid update. Error: ' + ERROR_MESSAGE() AS ErrorMessage
        END CATCH
    END
    ELSE
        SELECT 'No Matched Instructor ID';
END
```



```
GO
```

Uses

[dbo].[Instructor]

[dbo].[Update_Instructor_Course]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ins_id	int	4
@crs_id	int	4
@newCrsId	int	4

SQL Script

```
CREATE PROCEDURE [dbo].[Update_Instructor_Course]
    @ins_id INT,
    @crs_id INT,
    @newCrsId INT
AS
BEGIN
    IF EXISTS (SELECT 1 FROM Instructor_Courses WHERE Instructor_ID = @ins_id AND
Course_ID = @crs_id)
    BEGIN
        UPDATE Instructor_Courses
        SET Course_ID = @newCrsId
        WHERE Course_ID = @crs_id AND Instructor_ID = @ins_id;

        SELECT 'Instructor Course updated successfully' AS Status;
    END
    ELSE
        SELECT 'Instructor Course not found' AS Status;
END
GO
```

Uses

[dbo].[Instructor_Courses]

[dbo].[Update_Question]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@id	int	4
@Type	varchar(3)	3
@QuestionText	varchar(200)	200
@ModelAnswer	varchar(200)	200
@CourseID	int	4

SQL Script

```
--3-Update Question ------------
create proc [dbo].[Update_Question]
@id int ,
@Type VARCHAR(3),
@QuestionText VARCHAR(200),
@ModelAnswer VARCHAR(200),
@CourseID INT
as
begin
set nocount on;
if exists(select q.Question_ID  from Question q where q.Question_ID=@id)
begin
begin try
    update Question
    set Question.Type=@Type,
        Question.Question=@QuestionText,
        Question.Model_Answer=@ModelAnswer,
        Question.Course_ID=@CourseID
    where
    Question.Question_ID=@id
end try
begin catch
    select ERROR_NUMBER() AS ErrorNumber,
```



```
        ERROR_MESSAGE() AS ErrorMessage,
        ERROR_PROCEDURE() AS ErrorProcedure,
        ERROR_LINE() AS ErrorLine;
    end catch
    end
    else
        begin
            SELECT 'Question Does not exist' as ErrorMessage
        end
    END


GO
```

Uses

[dbo].[Question]

[dbo].[Update_Student]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StudID	int	4
@FirstName	varchar(15)	15
@SecondName	varchar(15)	15
@Email	varchar(50)	50
@Password	varchar(50)	50
@DeptID	int	4

SQL Script

```
create PROCEDURE [dbo].[Update_Student]
    @StudID INT,
    @FirstName VARCHAR(15) = 'NA',
    @SecondName VARCHAR(15) = 'NA',
    @Email VARCHAR(50) = 'NA',
    @Password VARCHAR(50) = 'NA',
    @DeptID INT = -1
AS
BEGIN
    IF EXISTS (SELECT Student_ID FROM Student WHERE Student_ID = @StudID)
    BEGIN
        BEGIN TRY
            IF @FirstName != 'NA'
                UPDATE Student SET First_Name = @FirstName WHERE Student_ID = @StudID

            IF @SecondName != 'NA'
                UPDATE Student SET Second_Name = @SecondName WHERE Student_ID = @StudID

            IF @Email != 'NA'
                UPDATE Student SET Email = @Email WHERE Student_ID = @StudID

            IF @Password != 'NA'
                UPDATE Student SET [Password] = @Password WHERE Student_ID = @StudID
```



```
            IF @DeptID != -1
                UPDATE Student SET Dept_ID = @DeptID WHERE Student_ID = @StudID
        END TRY

        BEGIN CATCH
            SELECT 'There Is An Error Happened!'
        END CATCH
    END

    ELSE SELECT 'No Student Found!';
END
GO
```

Uses

[dbo].[Student]

[dbo].[Update_Student_Answer]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ExamID	int	4
@QuID	int	4
@StudID	int	4
@StudAns	varchar(200)	200

SQL Script

```
-------------------------------------UPDATE----------------------------------------
CREATE PROCEDURE [dbo].[Update_Student_Answer]
    @ExamID INT,
    @QuID INT,
    @StudID INT,
    @StudAns varchar(200)
AS
BEGIN
    IF EXISTS (SELECT Student_Answer FROM Studend_Answer WHERE Exam_ID = @ExamID AND
Question_ID = @QuID AND Student_ID = @StudID)
    BEGIN
        BEGIN TRY
            UPDATE Student_Answer
            SET Student_Answer = @StudAns
            WHERE Exam_ID = @ExamID
            AND Question_ID = @QuID
            AND Student_ID = @StudID
        END TRY
        BEGIN CATCH
            SELECT 'There Is An Error Happened!'
        END CATCH
    END
    ELSE SELECT 'There is no Student Answer Matched!'
END
GO
```


Uses

[dbo].[Student_Answer]

[dbo].[Update_Topic_Name]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Course_ID	int	4
@OldTopicName	varchar(50)	50
@NewTopicName	varchar(50)	50

SQL Script

```
--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-- Update the topic name
CREATE PROCEDURE [dbo].[Update_Topic_Name]
    @Course_ID INT,
    @OldTopicName VARCHAR(50),
    @NewTopicName VARCHAR(50)
AS
BEGIN
    BEGIN TRY
        UPDATE Topic
        SET Topic_Name = @NewTopicName
        WHERE Course_ID = @Course_ID
          AND Topic_Name = @OldTopicName;

        PRINT 'Topic name updated successfully.';
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred while updating topic name.';
    END CATCH
END;
GO
```

Uses

[dbo].[Topic]

Users

Objects

Name
dbo
guest

dbo

Properties

Property	Value
Type	SqlUser
Login Name	sa
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
GO
```

guest

Properties

Property	Value
Type	SqlUser
Default Schema	guest

SQL Script

```
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