EXAMINATION SYSTEM

YEARO_EXAM_SYSTEM

Server	Mu-Atef\SQLEXPRESS
Author	dev-muatef
Created	BE S2ILE WIT17 OUR APPLICATION
File Path	C:\Users\dev-muatef\Documents\My Database Documentation\EXAMINATION-SYSTEM-2025-02-04T17-38-45.pdf

EXAMINATION SYSTEM APPLICATION

Table of Contents

Mu-Atef\SQLEXPRESS

Databases (1)

- YEARO_EXAM_SYSTEM

Server Properties

Property	Value
Product	Microsoft SQL Server
Version	16.0.1135.2
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	16263
Is Clustered	False
Root Directory	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL
Collation	SQL_Latin1_General_CP1_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)

- YEARO_EXAM_SYSTEM

YEARO_EXAM_SYSTEM Database

Files

Name	Type	Size	Maxsize	Autogrowth	File Name
YEARO Exam System	Data	72.00 MB	unlimited	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\YEARO-Exam-System\YEARO Exam System.mdf
YEARO Exam System_log	Log	72.00 MB	2048.00 GB	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\YEARO-Exam-System\YEARO Exam System_log.ldf

Tables

Objects

Name
dbo.Answer_Exam
dbo.Choice
dbo.Course
dbo.Department
dbo.Exam
dbo.Ins_Crs
dbo.Instructor
dbo.Intake
dbo.Person
dbo.Question
dbo.Std_Crs
dbo.Student
dbo.Student_Exam
dbo.Topic

[dbo].[Answer_Exam]

Properties

Property	Value
Row Count (~)	106
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	11:03:59 PM Sunday, February 2, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Default
PK FK C	ExamID	int	4	NOT NULL	
PK FK C	QID	int	4	NOT NULL	
PK FK C	StdID	int	4	NOT NULL	
	Answer	int	4	NULL allowed	((0))

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Answer_E__BB8BE9754E78CC28	ExamID, QID, StdID	True

Foreign Keys

Name	Delete	Columns
FK__Answer_Ex__ExamI__09746778	Cascade	ExamID->[dbo].[Exam].[ID]
FK__Answer_Exam__QID__0A688BB1		QID->[dbo].[Question].[ID]
FK__Answer_Ex__StdID__0B5CAFEA	Cascade	StdID->[dbo].[Student].[StdID]

SQL Script

```
CREATE TABLE [dbo].[Answer_Exam]
(
[ExamID] [int] NOT NULL,
[QID] [int] NOT NULL,
[StdID] [int] NOT NULL,
[Answer] [int] NULL CONSTRAINT [DF__Answer_Ex__Answe__0880433F] DEFAULT ((0))
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Answer_Exam] ADD CONSTRAINT [PK__Answer_E__BB8BE9754E78CC28] PRIMARY KEY
```

```
CLUSTERED ([ExamID], [QID], [StdID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Answer_Exam] ADD CONSTRAINT [FK__Answer_Ex__ExamI__09746778] FOREIGN KEY
([ExamID]) REFERENCES [dbo].[Exam] ([ID]) ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Answer_Exam] ADD CONSTRAINT [FK__Answer_Exam__QID__0A688BB1] FOREIGN KEY
([QID]) REFERENCES [dbo].[Question] ([ID])
GO
ALTER TABLE [dbo].[Answer_Exam] ADD CONSTRAINT [FK__Answer_Ex__StdID__0B5CAFEA] FOREIGN KEY
([StdID]) REFERENCES [dbo].[Student] ([StdID]) ON DELETE CASCADE
GO
```

Uses

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

Used By

[dbo].[generate_exam]
[dbo].[sp_calc_student_grade]
[dbo].[sp_GET_exam_question_choice]
[dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER]
[dbo].[sp_std_exam_answers]
[dbo].[sp_submit_exam_answer]

[dbo].[Choice]

Properties

Property	Value
Collation	Latin1_General_CI_AS
Row Count (~)	66
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK C	QID	int	4	NOT NULL
PK C	Choice	int	4	NOT NULL
	Body	nvarchar(max)	max	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Choice__387F3533C281229A	QID, Choice	True

Check Constraints

Name	On Column	Constraint
CK__Choice__Choice__04AFB25B	Choice	([Choice]>=(1) AND [Choice]<=(4))

Foreign Keys

Name	Delete	Columns
FK__Choice__QID__05A3D694	Cascade	QID->[dbo].[Question].[ID]

SQL Script

```
CREATE TABLE [dbo].[Choice]
(
[QID] [int] NOT NULL,
[Choice] [int] NOT NULL,
[Body] [nvarchar] (max) COLLATE Latin1_General_CI_AS NOT NULL
```

```
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Choice] ADD CONSTRAINT [CK__Choice__Choice__04AFB25B] CHECK (([Choice]>=(1) AND [Choice]<=(4)))
GO
ALTER TABLE [dbo].[Choice] ADD CONSTRAINT [PK__Choice__387F3533C281229A] PRIMARY KEY CLUSTERED ([QID], [Choice]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Choice] ADD CONSTRAINT [FK__Choice__QID__05A3D694] FOREIGN KEY ([QID]) REFERENCES [dbo].[Question] ([ID]) ON DELETE CASCADE
GO
```

Uses

[dbo].[Question]

Used By

[dbo].[pro_insert_question_choice]
[dbo].[sp_DeleteCourse]
[dbo].[sp_GET_exam_question_choice]
[dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER]
[dbo].[sp_std_exam_answers]
[dbo].[view_allQuestions_bycourse]
[dbo].[view_allQuestionsByAdmin]

[dbo].[Course]

Properties

Property	Value
Collation	Latin1_General_CI_AS
Row Count (~)	5
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK	ID	int	4	NOT NULL	1 - 1
	Name	nvarchar(100)	200	NOT NULL	
	Hours	int	4	NULL allowed	

Indexes

Key	Name	Key Columns	Unique
PK	PK__Course__3214EC27F25AE29F	ID	True

Check Constraints

Name	On Column	Constraint
CK__Course__Hours__69FBBC1F	Hours	([hours]>=(3) AND [hours]<=(100))

SQL Script

```
CREATE TABLE [dbo].[Course]
(
[ID] [int] NOT NULL IDENTITY(1, 1),
[Name] [nvarchar] (100) COLLATE Latin1_General_CI_AS NOT NULL,
[Hours] [int] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Course] ADD CONSTRAINT [CK__Course__Hours__69FBBC1F] CHECK (([hours]>=(3) AND
[hours]<=(100)))
GO
ALTER TABLE [dbo].[Course] ADD CONSTRAINT [PK__Course__3214EC27F25AE29F] PRIMARY KEY CLUSTERED
([ID]) ON [PRIMARY]
```

```
GO
```

Used By

[dbo].[Exam]
[dbo].[Ins_Crs]
[dbo].[Question]
[dbo].[Std_Crs]
[dbo].[Topic]
[dbo].[delete_course_questions]
[dbo].[delete_course_specific_question]
[dbo].[generate_exam]
[dbo].[GetCoursesByInstructor]
[dbo].[InsertInstructorCourse]
[dbo].[InsertStudentCourse]
[dbo].[ListStudentsForCourse]
[dbo].[ModifyCourseInstructor]
[dbo].[ModifyInstructorCourse]
[dbo].[pro_insert_question_choice]
[dbo].[sp_AddCourse]
[dbo].[sp_AddTopic]
[dbo].[sp_ChangeTopicCourse]
[dbo].[sp_DeleteCourse]
[dbo].[sp_DeleteTopic]
[dbo].[sp_get_all_courses]
[dbo].[sp_GetAllCoursesWithTopics]
[dbo].[sp_GetCourse]
[dbo].[sp_std_courses_grade]
[dbo].[sp_std_courses_instructor]
[dbo].[sp_submit_exam_answer]
[dbo].[sp_UpdateCourseHour]
[dbo].[sp_UpdateCourseName]
[dbo].[view_allQuestions_bycourse]
[dbo].[ViewAllInstructorCourse]
[dbo].[ViewCoursesByStudent]
[dbo].[ViewStudentCourse]

[dbo].[Department]

Properties

Property	Value
Collation	Latin1_General_CI_AS
Row Count (~)	3
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	ID	int	4	NOT NULL	1 - 1
	Name	nvarchar(100)	200	NOT NULL	
	Describe	nvarchar(255)	510	NULL allowed	

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Departme__3214EC2739B88E39	ID	True

SQL Script

```
CREATE TABLE [dbo].[Department]
(
[ID] [int] NOT NULL IDENTITY(1, 1),
[Name] [nvarchar] (100) COLLATE Latin1_General_CI_AS NOT NULL,
[Describe] [nvarchar] (255) COLLATE Latin1_General_CI_AS NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Department] ADD CONSTRAINT [PK__Departme__3214EC2739B88E39] PRIMARY KEY
CLUSTERED ([ID]) ON [PRIMARY]
GO
```

Used By

[dbo].[Person]
[dbo].[deleteAllDepartments]
[dbo].[DeleteSpecificDepartment]
[dbo].[InsertDepartment]

[dbo].[sp_add_person]
[dbo].[sp_std_info_depart]
[dbo].[updateDepartmentName]
[dbo].[ViewADepartmentsInfo]
[dbo].[ViewSpecificDepartmentInfo]

[dbo].[Exam]

Properties

Property	Value
Collation	Latin1_General_CI_AS
Row Count (~)	18
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	11:04:00 PM Sunday, February 2, 2025

Columns

Key	Name	Data Type	Persisted	Computed	Max Length (Bytes)	Nullability	Identity
PK C	ID	int			4	NOT NULL	1 - 1
	Name	nvarchar(100)			200	NULL allowed	
	StartDate	datetime			8	NOT NULL	
	EndDate	datetime	True	True	8	NULL allowed	
FK	CrsID	int			4	NOT NULL	

Computed columns

Name	Column definition
EndDate	(dateadd(hour,(1),[startdate]))

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Exam__3214EC271FDDC7DF	ID	True

Foreign Keys

Name	Delete	Columns
FK__Exam__CrsID__7755B73D	Cascade	CrsID->[dbo].[Course].[ID]

SQL Script

```
CREATE TABLE [dbo].[Exam]
(
```

```
[ID] [int] NOT NULL IDENTITY(1, 1),
[Name] [nvarchar] (100) COLLATE Latin1_General_CI_AS NULL,
[StartDate] [datetime] NOT NULL,
[EndDate] AS (dateadd(hour,(1),[startdate])) PERSISTED,
[CrsID] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [PK__Exam__3214EC271FDDC7DF] PRIMARY KEY CLUSTERED ([ID])
ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [FK__Exam__CrsID__7755B73D] FOREIGN KEY ([CrsID])
REFERENCES [dbo].[Course] ([ID]) ON DELETE CASCADE
GO
```

Uses

[dbo].[Course]

Used By

[dbo].[Answer_Exam]
[dbo].[Student_Exam]
[dbo].[deleteAllExams]
[dbo].[DeleteSpecificExam]
[dbo].[generate_exam]
[dbo].[sp_GET_exam_question_choice]
[dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER]
[dbo].[sp_std_courses_grade]
[dbo].[sp_submit_exam_answer]
[dbo].[updateExamStartDate]
[dbo].[ViewExamsInfo]
[dbo].[ViewSpecificExamInfo]

[dbo].[Ins_Crs]

Properties

Property	Value
Row Count (~)	9
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

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

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Ins_Crs__52BC6EE4194535C1	InsID, CrsID	True

Foreign Keys

Name	Delete	Columns
FK__Ins_Crs__CrsID__74794A92	Cascade	CrsID->[dbo].[Course].[ID]
FK__Ins_Crs__InsID__73852659	Cascade	InsID->[dbo].[Instructor].[InsID]

SQL Script

```
CREATE TABLE [dbo].[Ins_Crs]
(
[InsID] [int] NOT NULL,
[CrsID] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Ins_Crs] ADD CONSTRAINT [PK__Ins_Crs__52BC6EE4194535C1] PRIMARY KEY CLUSTERED
([InsID], [CrsID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Ins_Crs] ADD CONSTRAINT [FK__Ins_Crs__CrsID__74794A92] FOREIGN KEY ([CrsID])
REFERENCES [dbo].[Course] ([ID]) ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Ins_Crs] ADD CONSTRAINT [FK__Ins_Crs__InsID__73852659] FOREIGN KEY ([InsID])
REFERENCES [dbo].[Instructor] ([InsID]) ON DELETE CASCADE
```

```
GO
```

Uses

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

Used By

[dbo].[delete_course_questions]
[dbo].[delete_course_specific_question]
[dbo].[deleteAllInstructorCourseData]
[dbo].[DeleteInstructorAssignedToCourse]
[dbo].[DeleteInstructorCourses]
[dbo].[DeleteSpecificExam]
[dbo].[generate_exam]
[dbo].[GetCoursesByInstructor]
[dbo].[InsertInstructorCourse]
[dbo].[ModifyCourseInstructor]
[dbo].[ModifyInstructorCourse]
[dbo].[pro_insert_question_choice]
[dbo].[sp_GET_exam_question_choice]
[dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER]
[dbo].[sp_GetCourse]
[dbo].[sp_std_courses_instructor]
[dbo].[updateExamStartDate]
[dbo].[view_allQuestions_bycourse]
[dbo].[ViewAllInstructorCourse]
[dbo].[ViewSpecificExamInfo]

[dbo].[Instructor]

Properties

Property	Value
Row Count (~)	4
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK C	InsID	int	4	NOT NULL
	Hiredate	date	3	NULL allowed

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Instruct__9D104D8F0226DAF2	InsID	True

Foreign Keys

Name	Delete	Columns
FK__Instructo__InsID__671F4F74	Cascade	InsID->[dbo].[Person].[ID]

SQL Script

```
CREATE TABLE [dbo].[Instructor]
(
[InsID] [int] NOT NULL,
[Hiredate] [date] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Instructor] ADD CONSTRAINT [PK__Instruct__9D104D8F0226DAF2] PRIMARY KEY
CLUSTERED ([InsID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Instructor] ADD CONSTRAINT [FK__Instructo__InsID__671F4F74] FOREIGN KEY ([Ins-
ID]) REFERENCES [dbo].[Person] ([ID]) ON DELETE CASCADE
GO
```

Uses

[dbo].[Person]

Used By

[dbo].[Ins_Crs]
[dbo].[InsertInstructorCourse]
[dbo].[ModifyCourseInstructor]
[dbo].[ModifyInstructorCourse]
[dbo].[sp_add_person]
[dbo].[sp_get_all_instructors]
[dbo].[sp_get_instructor_byID]

[dbo].[Intake]

Properties

Property	Value
Collation	Latin1_General_CI_AS
Row Count (~)	4
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

Key	Name	Data Type	Persisted	Computed	Max Length (Bytes)	Nullability	Identity
PK C	ID	int			4	NOT NULL	1 - 1
	Name	nvarchar(100)			200	NOT NULL	
	StartDate	date			3	NOT NULL	
	EndDate	date	True	True	3	NULL allowed	

Computed columns

Name	Column definition
EndDate	(dateadd(month,(12),[startdate]))

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Intake__3214EC272521E11E	ID	True

Check Constraints

Name	Constraint
DateCheck	([StartDate]<[EndDate])

SQL Script

```
CREATE TABLE [dbo].[Intake]
(
[ID] [int] NOT NULL IDENTITY(1, 1),
```

```
[Name] [nvarchar] (100) COLLATE Latin1_General_CI_AS NOT NULL,
[StartDate] [date] NOT NULL,
[EndDate] AS (dateadd(month,(12),[startdate])) PERSISTED
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Intake] ADD CONSTRAINT [DateCheck] CHECK (([StartDate]<[EndDate]))
GO
ALTER TABLE [dbo].[Intake] ADD CONSTRAINT [PK__Intake__3214EC272521E11E] PRIMARY KEY CLUSTERED
([ID]) ON [PRIMARY]
GO
```

Used By

[dbo].[Student]
[dbo].[deleteIntake]
[dbo].[InsertIntake]
[dbo].[sp_add_person]
[dbo].[updateIntakeName]
[dbo].[updateIntakeNameAndDate]
[dbo].[updateIntakeStartDate]
[dbo].[ViewAIntakesInfo]
[dbo].[ViewSpecificIntakeInfo]

[dbo].[Person]

Properties

Property	Value
Collation	Latin1_General_CI_AS
Row Count (~)	19
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
	ID	int	4	NOT NULL	1 - 1
	Name	nvarchar(100)	200	NOT NULL	
	Email	nvarchar(255)	510	NOT NULL	
	NID	nvarchar(14)	28	NOT NULL	
	Address	nvarchar(255)	510	NOT NULL	
	Gender	char(1)	1	NULL allowed	
	Salary	decimal(18,2)	9	NULL allowed	
	DOB	date	3	NULL allowed	
	Phone	nvarchar(15)	30	NULL allowed	
	Role	nvarchar(30)	60	NOT NULL	
	Password	nvarchar(255)	510	NOT NULL	
	DeptID	int	4	NULL allowed	

Indexes

Key	Name	Key Columns	Unique
	PK__Person__3214EC27B685C176	ID	True
	UQ__Person__A9D10534B187647E	Email	True
	UQ__Person__C7DEC33287550E40	NID	True

Check Constraints

Name	On Column	Constraint
CK__Person__Email__5AB9788F	Email	([Email] like '%___@___%.__%')

CK__Person__Gender__5CA1C101	Gender	([Gender]='M' OR [Gender]='F')
CK__Person__Role__5E8A0973	Role	([Role]='Instructor' OR [Role]='Student' OR [Role]='Admin')
CK__Person__DOB__5D95E53A	DOB	(datediff(year,[DOB],getdate())>=(22) AND datediff(year,[DOB],getdate())<=(70))
CK__Person__NID__5BAD9CC8	NID	(len([NID])=(14) AND [NID] like '%[0-9]%')
CK__Person__Password__5F7E2DAC	Password	(len([Password])>=(3))

Foreign Keys

Name	Delete	Columns
FK__Person__DeptID__607251E5	SetNull	DeptID->[dbo].[Department].[ID]

SQL Script

```
CREATE TABLE [dbo].[Person]
(
[ID] [int] NOT NULL IDENTITY(1, 1),
[Name] [nvarchar] (100) COLLATE Latin1_General_CI_AS NOT NULL,
[Email] [nvarchar] (255) COLLATE Latin1_General_CI_AS NOT NULL,
[NID] [nvarchar] (14) COLLATE Latin1_General_CI_AS NOT NULL,
[Address] [nvarchar] (255) COLLATE Latin1_General_CI_AS NOT NULL,
[Gender] [char] (1) COLLATE Latin1_General_CI_AS NULL,
[Salary] [decimal] (18, 2) NULL,
[DOB] [date] NULL,
[Phone] [nvarchar] (15) COLLATE Latin1_General_CI_AS NULL,
[Role] [nvarchar] (30) COLLATE Latin1_General_CI_AS NOT NULL,
[Password] [nvarchar] (255) COLLATE Latin1_General_CI_AS NOT NULL,
[DeptID] [int] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [CK__Person__Email__5AB9788F] CHECK (([Email] like
'%___@___%.__%'))
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [CK__Person__Gender__5CA1C101] CHECK (([Gender]='M' OR
[Gender]='F'))
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [CK__Person__Role__5E8A0973] CHECK
(([Role]='Instructor' OR [Role]='Student' OR [Role]='Admin'))
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [CK__Person__DOB__5D95E53A] CHECK
((datediff(year,[DOB],getdate())>=(22) AND datediff(year,[DOB],getdate())<=(70)))
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [CK__Person__NID__5BAD9CC8] CHECK ((len([NID])=(14) AND
[NID] like '%[0-9]%'))
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [CK__Person__Password__5F7E2DAC] CHECK
((len([Password])>=(3)))
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [PK__Person__3214EC27B685C176] PRIMARY KEY CLUSTERED
([ID]) ON [PRIMARY]
GO
```

```sql
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [UQ__Person__A9D10534B187647E] UNIQUE NONCLUSTERED
([Email]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [UQ__Person__C7DEC33287550E40] UNIQUE NONCLUSTERED
([NID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [FK__Person__DeptID__607251E5] FOREIGN KEY ([DeptID])
REFERENCES [dbo].[Department] ([ID]) ON DELETE SET NULL
GO
```

Uses

[dbo].[Department]

Used By

[dbo].[Instructor]
[dbo].[Student]
[dbo].[delete_course_questions]
[dbo].[delete_course_specific_question]
[dbo].[DeleteSpecificExam]
[dbo].[generate_exam]
[dbo].[GetCoursesByInstructor]
[dbo].[InsertIntake]
[dbo].[ListStudentsForCourse]
[dbo].[PERSON_Update]
[dbo].[pro_insert_question_choice]
[dbo].[sp_add_person]
[dbo].[sp_delete_person]
[dbo].[sp_get_all_instructors]
[dbo].[sp_get_all_persons]
[dbo].[sp_get_all_students]
[dbo].[sp_GET_exam_question_choice]
[dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER]
[dbo].[sp_get_instructor_byID]
[dbo].[sp_get_person_role]
[dbo].[sp_get_student_byID]
[dbo].[sp_GetCourse]
[dbo].[sp_std_info_depart]
[dbo].[sp_submit_exam_answer]
[dbo].[updateExamStartDate]
[dbo].[view_allQuestions_bycourse]
[dbo].[ViewAllInstructorCourse]
[dbo].[ViewCoursesByStudent]
[dbo].[ViewSpecificExamInfo]
[dbo].[ViewStudentCourse]
[instructor].[GET_PERSON]

[dbo].[Question]

Properties

Property	Value
Collation	Latin1_General_CI_AS
Row Count (~)	19
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity	Default
PK	ID	int	4	NOT NULL	1 - 1	
	Body	nvarchar(max)	max	NOT NULL		
	Type	nvarchar(10)	20	NOT NULL		
	Degree	int	4	NOT NULL		
	CorrectChoice	int	4	NOT NULL		
FK	CrsID	int	4	NOT NULL		
	isDeleted	bit	1	NULL allowed		((0))

Indexes

Key	Name	Key Columns	Unique
PK	PK__Question__3214EC27EF5AB5C9	ID	True

Check Constraints

Name	On Column	Constraint
CK__Question__Correc__7FEAFD3E	CorrectChoice	([CorrectChoice]>=(1) AND [CorrectChoice]<=(4))
CK__Question__Degree__7EF6D905	Degree	([Degree]>=(1) AND [Degree]<=(5))
CK__Question__Type__7E02B4CC	Type	([Type]='MCQ' OR [Type]='T/F')

Foreign Keys

Name	Delete	Columns
FK__Question__CrsID__01D345B0	Cascade	CrsID->[dbo].[Course].[ID]

SQL Script

```
CREATE TABLE [dbo].[Question]
(
[ID] [int] NOT NULL IDENTITY(1, 1),
[Body] [nvarchar] (max) COLLATE Latin1_General_CI_AS NOT NULL,
[Type] [nvarchar] (10) COLLATE Latin1_General_CI_AS NOT NULL,
[Degree] [int] NOT NULL,
[CorrectChoice] [int] NOT NULL,
[CrsID] [int] NOT NULL,
[isDeleted] [bit] NULL CONSTRAINT [DF__Question__isDele__00DF2177] DEFAULT ((0))
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [CK__Question__Correc__7FEAFD3E] CHECK (([Correct-
Choice]>=(1) AND [CorrectChoice]<=(4)))
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [CK__Question__Degree__7EF6D905] CHECK
(([Degree]>=(1) AND [Degree]<=(5)))
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [CK__Question__Type__7E02B4CC] CHECK (([Type]='MCQ'
OR [Type]='T/F'))
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [PK__Question__3214EC27EF5AB5C9] PRIMARY KEY
CLUSTERED ([ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [FK__Question__CrsID__01D345B0] FOREIGN KEY ([CrsID])
REFERENCES [dbo].[Course] ([ID]) ON DELETE CASCADE
GO
```

Uses

[dbo].[Course]

Used By

[dbo].[Answer_Exam]
[dbo].[Choice]
[dbo].[delete_course_questions]
[dbo].[delete_course_specific_question]
[dbo].[generate_exam]
[dbo].[pro_insert_question_choice]
[dbo].[sp_calc_student_grade]
[dbo].[sp_DeleteCourse]
[dbo].[sp_GET_exam_question_choice]
[dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER]
[dbo].[sp_std_exam_answers]
[dbo].[view_allQuestions_bycourse]
[dbo].[view_allQuestionsByAdmin]

[dbo].[Std_Crs]

Properties

Property	Value
Row Count (~)	12
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

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

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Std_Crs__9A708D54205D8DCA	StdID, CrsID	True

Foreign Keys

Name	Delete	Columns
FK__Std_Crs__CrsID__70A8B9AE	Cascade	CrsID->[dbo].[Course].[ID]
FK__Std_Crs__StdID__6FB49575	Cascade	StdID->[dbo].[Student].[StdID]

SQL Script

```
CREATE TABLE [dbo].[Std_Crs]
(
[StdID] [int] NOT NULL,
[CrsID] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Std_Crs] ADD CONSTRAINT [PK__Std_Crs__9A708D54205D8DCA] PRIMARY KEY CLUSTERED
([StdID], [CrsID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Std_Crs] ADD CONSTRAINT [FK__Std_Crs__CrsID__70A8B9AE] FOREIGN KEY ([CrsID])
REFERENCES [dbo].[Course] ([ID]) ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Std_Crs] ADD CONSTRAINT [FK__Std_Crs__StdID__6FB49575] FOREIGN KEY ([StdID])
REFERENCES [dbo].[Student] ([StdID]) ON DELETE CASCADE
```

```
GO
```

Uses

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

Used By

[dbo].[DeleteStudentAndAssociatedCourses]
[dbo].[DeleteStudentCoureTable]
[dbo].[DropCourseAndEnrolledStudents]
[dbo].[generate_exam]
[dbo].[InsertStudentCourse]
[dbo].[ListStudentsForCourse]
[dbo].[sp_std_courses_instructor]
[dbo].[sp_submit_exam_answer]
[dbo].[ViewCoursesByStudent]
[dbo].[ViewStudentCourse]

[dbo].[Student]

Properties

Property	Value
Collation	Latin1_General_CI_AS
Row Count (~)	9
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	11:04:00 PM Sunday, February 2, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK C	StdID	int	4	NOT NULL
FK	IntakeID	int	4	NULL allowed
	College	nvarchar(255)	510	NULL allowed

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Student__55DCAE3FEA1B49D1	StdID	True

Foreign Keys

Name	Delete	Columns
FK__Student__College__634EBE90	SetNull	IntakeID->[dbo].[Intake].[ID]
FK__Student__StdID__6442E2C9	Cascade	StdID->[dbo].[Person].[ID]

SQL Script

```
CREATE TABLE [dbo].[Student]
(
[StdID] [int] NOT NULL,
[IntakeID] [int] NULL,
[College] [nvarchar] (255) COLLATE Latin1_General_CI_AS NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [PK__Student__55DCAE3FEA1B49D1] PRIMARY KEY CLUSTERED
([StdID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [FK__Student__College__634EBE90] FOREIGN KEY ([Intake-
```

```
ID]) REFERENCES [dbo].[Intake] ([ID]) ON DELETE SET NULL
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [FK__Student__StdID__6442E2C9] FOREIGN KEY ([StdID])
REFERENCES [dbo].[Person] ([ID]) ON DELETE CASCADE
GO
```

Uses

[dbo].[Intake]
[dbo].[Person]

Used By

[dbo].[Answer_Exam]
[dbo].[Std_Crs]
[dbo].[Student_Exam]
[dbo].[InsertStudentCourse]
[dbo].[sp_add_person]
[dbo].[sp_get_all_students]
[dbo].[sp_get_student_byID]
[dbo].[sp_std_info_depart]

[dbo].[Student_Exam]

Properties

Property	Value
Row Count (~)	22
Created	11:03:59 PM Sunday, February 2, 2025
Last Modified	11:04:41 PM Sunday, February 2, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Default
PK FK C	StdID	int	4	NOT NULL	
PK FK C	ExamID	int	4	NOT NULL	
	Grade	int	4	NOT NULL	((0))

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Student___374BFC25A938DE9C	StdID, ExamID	True

Foreign Keys

Name	Delete	Columns
FK__Student_E__ExamI__7B264821	Cascade	ExamID->[dbo].[Exam].[ID]
FK__Student_E__StdID__7A3223E8	Cascade	StdID->[dbo].[Student].[StdID]

SQL Script

```
CREATE TABLE [dbo].[Student_Exam]
(
[StdID] [int] NOT NULL,
[ExamID] [int] NOT NULL,
[Grade] [int] NOT NULL CONSTRAINT [DF_grade] DEFAULT ((0))
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Exam] ADD CONSTRAINT [PK__Student___374BFC25A938DE9C] PRIMARY KEY
CLUSTERED ([StdID], [ExamID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Exam] ADD CONSTRAINT [FK__Student_E__ExamI__7B264821] FOREIGN KEY
([ExamID]) REFERENCES [dbo].[Exam] ([ID]) ON DELETE CASCADE
```

```
GO
ALTER TABLE [dbo].[Student_Exam] ADD CONSTRAINT [FK__Student_E__StdID__7A3223E8] FOREIGN KEY
([StdID]) REFERENCES [dbo].[Student] ([StdID]) ON DELETE CASCADE
GO
```

Uses

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

Used By

[dbo].[generate_exam]
[dbo].[sp_calc_student_grade]
[dbo].[sp_std_courses_grade]

[dbo].[Topic]

Properties

Property	Value
Collation	Latin1_General_CI_AS
Row Count (~)	8
Created	4:29:49 PM Saturday, February 1, 2025
Last Modified	4:29:49 PM Saturday, February 1, 2025

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	ID	int	4	NOT NULL	1 - 1
	Name	nvarchar(100)	200	NOT NULL	
FK	CrsID	int	4	NOT NULL	

Indexes

Key	Name	Key Columns	Unique
PK C	PK__Topic__3214EC2718D5F5B4	ID	True

Foreign Keys

Name	Delete	Columns
FK__Topic__CrsID__6CD828CA	Cascade	CrsID->[dbo].[Course].[ID]

SQL Script

```
CREATE TABLE [dbo].[Topic]
(
[ID] [int] NOT NULL IDENTITY(1, 1),
[Name] [nvarchar] (100) COLLATE Latin1_General_CI_AS NOT NULL,
[CrsID] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Topic] ADD CONSTRAINT [PK__Topic__3214EC2718D5F5B4] PRIMARY KEY CLUSTERED
([ID]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Topic] ADD CONSTRAINT [FK__Topic__CrsID__6CD828CA] FOREIGN KEY ([CrsID])
REFERENCES [dbo].[Course] ([ID]) ON DELETE CASCADE
```

```
GO
```

Uses

[dbo].[Course]

Used By

[dbo].[sp_AddTopic]
[dbo].[sp_ChangeTopicCourse]
[dbo].[sp_DeleteCourse]
[dbo].[sp_DeleteTopic]
[dbo].[sp_GetAllCoursesWithTopics]
[dbo].[sp_GetCourse]
[dbo].[sp_GetCourse_ITS_Topics]
[dbo].[sp_UpdateTopicName]

Stored Procedures

Objects

Name
dbo.delete_course_questions *This stored procedure, delete_course_questions, soft deletes questions associated with a specific course by setting their isDeleted flag to 1. It first verifies if the course exists and checks if the current user is the instructor for the course. If the user is not authorized or the course is not found, appropriate messages are displayed.*
dbo.delete_course_specific_question *This stored procedure, delete_course_specific_question, soft deletes a specific question (by setting isDeleted to 1) within a given course. It verifies if the course exists, checks if the current user is the instructor for the course, and ensures the question belongs to the specified course. If any condition fails, appropriate messages are displayed.*
dbo.deleteAllDepartments *This stored procedure deleteAllDepartments deletes all records from the department table and prints a confirmation message. It is encrypted to prevent viewing its definition.*
dbo.deleteAllExams *The deleteAllExams stored procedure deletes all records from the exam table and prints a confirmation message. It is encrypted to protect its definition.*
dbo.deleteAllInstructorCourseData *The deleteAllInstructorCourseData stored procedure deletes all records from the Ins_Crs table and prints a confirmation message. It is encrypted to secure its definition.*
dbo.DeleteInstructorAssignedToCourse *The DeleteInstructorAssignedToCourse stored procedure removes a specific course assignment for an instructor from the Ins_Crs table. It validates inputs, checks for existence, and handles errors with a TRY...CATCH block. The procedure is encrypted to protect its definition.*
dbo.DeleteInstructorCourses *The DeleteInstructorCourses stored procedure deletes all course assignments for a given instructor from the Ins_Crs table. It validates input, checks for existence, and handles errors using a TRY...CATCH block. The procedure is encrypted to secure its definition.*
dbo.deleteIntake *The deleteIntake stored procedure deletes a specific intake from the Intake table based on its name. It first checks for existence before deletion and prints a success or failure message accordingly.*
dbo.DeleteLoginAndUser *The DeleteLoginAndUser stored procedure removes a SQL Server login and its corresponding database user if they exist. It first checks for their existence and then executes dynamic SQL to drop them securely.*
dbo.DeleteSpecificDepartment *The DeleteSpecificDepartment stored procedure deletes a department based on its ID. It checks for null or empty input, verifies existence before deletion, and prints an appropriate message. The procedure is encrypted to secure its definition.*
dbo.DeleteSpecificExam *The DeleteSpecificExam stored procedure deletes an exam based on its ID. It verifies input, checks if the caller is an authorized instructor, and ensures the exam exists before deletion. The procedure is encrypted to secure its definition.*
dbo.DeleteStudentAndAssociatedCourses *The DeleteStudentAndAssociatedCourses stored procedure deletes the courses associated with a specific student based on their ID. It checks if the student exists, handles errors using TRY...CATCH, and prints appropriate messages. The procedure is encrypted to secure its definition.*
dbo.DeleteStudentCoureTable *The DeleteStudentCoureTable stored procedure deletes all records from the Std_Crs table. It uses a TRY...CATCH block to handle potential errors and prints a success or error message accordingly. The procedure is encrypted to secure its definition.*
dbo.DropCourseAndEnrolledStudents *The DropCourseAndEnrolledStudents stored procedure deletes a course and its associated enrolled students based on the provided course ID. It checks for null input, verifies the existence of the course, and handles errors using a TRY...CATCH block. The procedure is encrypted to secure its definition.*

dbo.generate_exam
The generate_exam stored procedure creates a new exam for a specific course. It ensures that the course exists and the user is the instructor, then generates the exam and randomly selects questions from the Question table. It also sets up student participation by inserting relevant data into the Answer_Exam and Student_Exam tables. The procedure uses a TRY...CATCH block for error handling and transactions to ensure data consistency. The procedure is encrypted to secure its definition.

dbo.GetCoursesByInstructor
The GetCoursesByInstructor stored procedure retrieves the courses taught by a specific instructor based on their ID. It checks for null or invalid input, verifies that the instructor exists, and returns details about the instructor and their courses. The procedure uses a TRY...CATCH block to handle potential errors and is encrypted to protect its definition.

dbo.InsertDepartment
The InsertDepartment stored procedure inserts a new department into the Department table if the department name does not already exist. It checks for empty or null input and handles errors using a TRY...CATCH block. The procedure prints appropriate success or error messages and is encrypted to protect its definition.

dbo.InsertInstructorCourse
The InsertInstructorCourse stored procedure assigns an instructor to a course by inserting a record into the Ins_Crs table. It validates the inputs, ensures that both the instructor and course exist, and checks if the assignment already exists. The procedure uses a TRY...CATCH block for error handling and prints appropriate success or error messages. It is encrypted to protect its definition.

dbo.InsertIntake
The InsertIntake stored procedure inserts a new intake record into the Intake table. It validates that the name is not empty, ensures the start date is not in the past, and checks if the user is an admin before proceeding. It uses a TRY...CATCH block for error handling and prints appropriate messages for success or failure.

dbo.InsertStudentCourse
The InsertStudentCourse stored procedure assigns a student to a course by inserting a record into the Std_Crs table. It validates the student and course IDs, checks if the assignment already exists, and ensures that both the student and course are valid. The procedure uses a TRY...CATCH block for error handling and prints appropriate messages for success or error. It is encrypted to secure its definition.

dbo.ListStudentsForCourse
The ListStudentsForCourse stored procedure retrieves a list of students enrolled in a specific course based on the provided course ID. It checks if the course exists and returns student details, including their ID, name, and the course name. If the course doesn't exist, it prints a message indicating so. The procedure is encrypted to secure its definition.

dbo.ModifyCourseInstructor
The ModifyCourseInstructor stored procedure allows you to update the instructor assigned to a specific course. It checks if the old and new instructor IDs, as well as the course ID, are valid. If the old instructor is assigned to the course, it will update the record to reflect the new instructor unless the new assignment already exists. The procedure ensures that all required conditions are met and handles errors with a TRY...CATCH block. It prints messages based on the outcome of the operation, providing feedback to the user.

dbo.ModifyInstructorCourse
This stored procedure updates the course assignment for an instructor. It validates input parameters for instructor and course existence, checks if the assignment exists, and performs the update. If any validation fails, it prints an appropriate message.

dbo.PERSON_Update
The PERSON_Update stored procedure updates a person's email in the Person table while ensuring the email is unique and the ID exists. It deletes the old login, updates the email, and then creates a new login and user with the updated email, preserving the person's role and password.

dbo.pro_insert_question_choice
The pro_insert_question_choice stored procedure inserts a question with choices into the Question and Choice tables based on the provided course name. It ensures that the user is the instructor for the course, validates the choices for MCQs, and handles both MCQ and true/false question types. If all conditions are met, it commits the transaction; otherwise, it rolls back.

dbo.sp_add_person
The sp_add_person stored procedure inserts a new person into the Person table, handling different roles (Student, Instructor, Admin). It checks if the provided department and intake IDs exist, creates the appropriate login and user, and adds them to relevant roles. The procedure commits the transaction if successful, or rolls it back in case of an error.

dbo.sp_AddCourse
The sp_AddCourse stored procedure adds a new course to the Course table after validating that the course hours are between 3 and 100. If the hours are outside this range, an error message is raised. If valid, the course is inserted into the table.

dbo.sp_AddTopic
The sp_AddTopic stored procedure adds a new topic to the Topic table, associating it with a specified course. It first checks if the course exists by name; if not, it prints an error message. If the course is found, the topic is inserted with the corresponding course ID.

dbo.sp_calc_student_grade

The sp_calc_student_grade stored procedure calculates a student's grade for a specific exam by comparing the answers with the correct choices. It computes the total grade, percentage, and updates the Student_Exam table with the student's grade. If the data is invalid or an error occurs, the transaction is rolled back.

dbo.sp_ChangeTopicCourse
The sp_ChangeTopicCourse stored procedure changes the course assignment of a specific topic. It checks if the topic and both courses exist, then updates the topic's course ID to the new course. If any of the conditions fail, it prints an appropriate error message.

dbo.sp_delete_person
The sp_delete_person stored procedure deletes a person from the Person table based on the provided person ID. It checks if the ID is valid and exists, then deletes the corresponding record and removes the associated login using the DeleteLoginAndUser procedure. If the ID is null or the person does not exist, it prints an appropriate message.

dbo.sp_DeleteCourse
The sp_DeleteCourse stored procedure deletes a course and its related data. It first checks if the course exists by name. If found, it deletes associated topics, question choices, and related questions before deleting the course itself. If the course does not exist, it prints an error message.

dbo.sp_DeleteTopic
The sp_DeleteTopic stored procedure deletes a topic from a specified course. It first checks if the topic and course exist. If they do, the topic is removed from the course. If either the topic or course is not found, an appropriate error message is printed.

dbo.sp_get_all_courses
The sp_get_all_courses stored procedure retrieves all records from the Course table, providing a list of all available courses.

dbo.sp_get_all_instructors
The sp_get_all_instructors stored procedure retrieves details of all instructors, including their personal information from the Person table and their hire date from the Instructor table, by performing an inner join between the two tables. The procedure is encrypted to secure its definition.

dbo.sp_get_all_persons
The sp_get_all_persons stored procedure retrieves all the records from the Person table, returning the complete details of all individuals stored in the database. The procedure is encrypted to secure its definition.

dbo.sp_get_all_students
The sp_get_all_students stored procedure retrieves all records from the Person table along with IntakeID and College details from the Student table, using an inner join on ID and StdID. The procedure is encrypted to protect its definition.

dbo.sp_GET_exam_question_choice
The sp_GET_exam_question_choice stored procedure retrieves the questions and choices for a specific exam identified by @examid. It checks if the user is an instructor or an admin before fetching the question and choice data. The procedure ensures the @examid is valid and returns choices for each question in the exam, or prints relevant messages for invalid inputs or access denial.

dbo.sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER
The sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER stored procedure retrieves questions and the student's answers for a specific exam identified by @examid and @stdID. It checks if the user has access rights (instructor or admin) and ensures that the exam and student exist. If valid, it returns the questions along with the student's answers from the Answer_Exam table.

dbo.sp_get_instructor_byID
The sp_get_instructor_byID stored procedure retrieves the details of an instructor based on the provided @insID. It checks if the @insID is not null and exists in the Instructor table. If valid, it returns the instructor's personal details along with their hire date; otherwise, it prints an error message indicating the instructor does not exist.

dbo.sp_get_person_role
The sp_get_person_role stored procedure retrieves the name and role of a person based on the provided @personID. It checks if the @personID is not null and exists in the Person table. If valid, it returns the person's name and role; otherwise, it prints an error message indicating the person does not exist.

dbo.sp_get_student_byID
The sp_get_student_byID stored procedure retrieves the details of a student based on the provided @stdID. It checks if the @stdID is not null and exists in the Student table. If valid, it returns the student's personal details along with their IntakeID and College; otherwise, it prints an error message indicating the student does not exist.

dbo.sp_GetAllCoursesWithTopics
The sp_GetAllCoursesWithTopics stored procedure retrieves a list of all courses along with their associated topics. It returns the course name, the number of hours for each course, and a concatenated list of topics (or "no topic" if none are associated). The procedure uses a left join between the Course and Topic tables, grouping by course name and hours.

dbo.sp_GetCourse

dbo.sp_GetCourse_ITS_Topics
The sp_GetCourse_ITS_Topics stored procedure retrieves a list of topic names associated with a specific course, identified by @CrsID. It selects the topic names from the Topic table where the CrsID matches the provided value. The procedure is executed by

passing a specific CrsID (e.g., 1 in this case).

dbo.sp_std_courses_grade
The sp_std_courses_grade stored procedure retrieves the grades of a student for completed exams. It takes @stdID as input and joins the Student_Exam, Exam, and Course tables to return the course name, exam name, and grade. It filters results by ensuring the exam's EndDate is before the current date, meaning only completed exams are included. The procedure is executed by passing a specific student ID (e.g., 3 in this case).

dbo.sp_std_courses_instructor
The sp_std_courses_instructor stored procedure retrieves the list of courses taught by a specific instructor, identified by @instruct-ID, along with the number of students enrolled in each course. It joins the Course, Ins_Crs, and Std_Crs tables to count the students per course. The procedure returns the course name and the corresponding student count, grouped by course name.

dbo.sp_std_exam_answers

dbo.sp_std_info_depart
The sp_std_info_depart stored procedure retrieves the personal information of students belonging to a specific department, identified by @deptID. It joins the Person, Department, and Student tables to return the student's details along with their IntakeID and College, filtered by the department and the role being 'student'.

dbo.sp_submit_exam_answer
The sp_submit_exam_answer stored procedure allows a student to submit an answer for a specific exam question. It performs several validation checks, including verifying the student's enrollment in the course, checking the exam's start and end dates, and ensuring the student's answer hasn't been submitted already. If all conditions are met, it updates the student's answer in the Answer_Exam table and calculates the student's grade. The procedure ensures data integrity by using transactions and rolling back if any validation fails or errors occur.

dbo.sp_UpdateCourseHour
The sp_UpdateCourseHour stored procedure updates the number of hours for a course specified by its Name. It first checks if the course exists in the Course table. If no course is found, it prints a message. If the specified @Hour is not between 3 and 100, an error is raised. Otherwise, it updates the course's hours with the provided value.

dbo.sp_UpdateCourseName
The sp_UpdateCourseName stored procedure updates the name of a course specified by @oldName to a new name provided as @newName. It first checks if the course exists in the Course table by searching for the @oldName. If the course is not found, it prints a message. Otherwise, it updates the course name with the new value.

dbo.sp_UpdateTopicName
The sp_UpdateTopicName stored procedure updates the name of a topic from @oldTopicName to @newTopicName. It checks if the topic with the given @oldTopicName exists in the Topic table. If the topic is found, it updates the topic's name. Otherwise, it prints a message indicating that no topic with the specified name was found.

dbo.updateDepartmentName
The updateDepartmentName stored procedure updates the name and description of a department based on the provided @DID (Department ID). It first checks if the @DID or @newName is null or empty. If any of these are invalid, it prints an error message. Then, it verifies if the department exists using the @DID. If the department exists, it updates the department's name and description; otherwise, it prints a message indicating the department doesn't exist.

dbo.updateExamStartDate
The updateExamStartDate stored procedure updates the start date of an exam identified by @EX_id to a new date provided by @newdate. It checks if the @EX_id is valid and whether the new start date is in the future. The procedure also ensures that the user has instructor-level access for the course associated with the exam. If the exam exists, the start date is updated; otherwise, it prints an error message. It includes error handling to manage potential issues during execution.

dbo.updateIntakeName
The updateIntakeName stored procedure updates the name of an intake from @oldName to @newName. It first checks if the @old-Name is not null or empty. Then, it verifies if the intake with the given @oldName exists in the Intake table. If the intake exists, it updates the name to @newName; otherwise, it prints an error message indicating the intake doesn't exist.

dbo.updateIntakeNameAndDate
The updateIntakeNameAndDate stored procedure updates both the name and the start date of an intake specified by @oldName. It first checks if the @oldName is not null or empty. Then, it verifies if an intake with the given name exists. If the intake is found, it starts a transaction to update the intake's name and start date to the provided values (@NewName and @New_date). If any error occurs during the process, it rolls back the transaction and prints an error message. If the intake name doesn't exist, it prints a message indicating so.

dbo.updateIntakeStartDate
The updateIntakeStartDate stored procedure updates the start date of an intake identified by @intake_Name to the provided @St_date. It first checks if the @intake_Name is not null or empty. Then, it verifies if the intake with the given name exists. If the intake exists, it updates the start date. Otherwise, it prints a message indicating the intake does not exist.

dbo.view_allQuestions_bycourse
The view_allQuestions_bycourse stored procedure retrieves all questions and their associated choices for a given course specified

by @CrsName. It first checks if the course exists and if the user has instructor-level access for that course. If the course is not found or the user is not an instructor for the course, an appropriate message is displayed. If the course exists and the user is authorized, the procedure returns the questions along with their possible choices (Choice1, Choice2, etc.) from the Question and Choice tables.
dbo.view_allQuestionsByAdmin *The view_allQuestionsByAdmin stored procedure retrieves all questions along with their associated choices (Choice1, Choice2, etc.) for all courses in the system, without any course or instructor restrictions. The procedure selects the question body and the possible choices from the Question and Choice tables and groups the results by question ID and body. The procedure is encrypted for security purposes.*
dbo.ViewADepartmentsInfo *The ViewADepartmentsInfo stored procedure retrieves all records from the Department table, displaying all available department details in the system. This procedure is encrypted for security purposes.*
dbo.ViewAIntakesInfo *The ViewAIntakesInfo stored procedure retrieves all records from the Intake table, displaying the details of all intakes in the system. This procedure is encrypted for security purposes.*
dbo.ViewAllInstructorCourse *The ViewAllInstructorCourse stored procedure retrieves details about all courses assigned to instructors. It returns the instructor's ID, name, course name, and course ID by joining the Ins_Crs, Person, and Course tables. This procedure is encrypted for security purposes.*
dbo.ViewCoursesByStudent *The ViewCoursesByStudent stored procedure retrieves courses associated with a specific student. If the student ID is provided, it checks if the student exists in the Std_Crs table. If the student exists, it returns their ID, name, the course ID, and course name. Otherwise, it prints an error message if the student ID is not found or if the ID is null.* *This procedure ensures that only valid student IDs are processed, and course details are fetched accordingly.*
dbo.ViewExamsInfo *The ViewExamsInfo stored procedure retrieves all records from the Exam table. It doesn't require any input parameters and will simply return all the columns of the Exam table when executed. This procedure is helpful for administrators or instructors who need to view the complete list of exams.*
dbo.ViewSpecificDepartmentInfo *The ViewSpecificDepartmentInfo procedure retrieves details of a department by its ID (@DID). It checks if the department exists, and if so, returns its information. If the ID is invalid or not found, an appropriate message is displayed. This is useful for administrators or users looking for specific department data.*
dbo.ViewSpecificExamInfo *The ViewSpecificExamInfo stored procedure retrieves details of a specific exam based on the provided @EX_ID (exam ID). It checks if the exam exists and if the user is authorized (i.e., the user is the instructor for the related course). If the exam ID is not valid or the user lacks permission, an appropriate message is displayed. Otherwise, the exam information is returned. This procedure is useful for instructors or administrators who need to view the details of a particular exam.*
dbo.ViewSpecificIntakeInfo *The ViewSpecificIntakeInfo stored procedure retrieves the details of a specific intake based on the provided @intake_Name. It checks if the intake name is valid and exists in the Intake table. If the intake name is not found or is empty, an appropriate message is displayed. Otherwise, the procedure returns all information for the matching intake. This procedure is useful for administrators or department staff who need to view details of a particular intake.*
dbo.ViewStudentCourse *The ViewStudentCourse stored procedure retrieves the list of students and the courses they are enrolled in. It provides information such as the student ID, student name, course ID, and course name. This procedure is useful for administrators or instructors who want to view the enrollment details of students in different courses. The query joins the Std_Crs, Person, and Course tables to return the relevant data.*
instructor.GET_PERSON

[dbo].[delete_course_questions]

MS_Description

This stored procedure, delete_course_questions, soft deletes questions associated with a specific course by setting their isDeleted flag to 1. It first verifies if the course exists and checks if the current user is the instructor for the course. If the user is not authorized or the course is not found, appropriate messages are displayed.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@CrsName	nvarchar(max)	max

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR

SQL Script

```
CREATE   PROCEDURE [dbo].[delete_course_questions]
@CrsName NVARCHAR(MAX)
As
begin

    declare @CrsID int;
    select @CrsID=id from Course where Name=@CrsName;

    IF @@ROWCOUNT = 0
       PRINT 'No course found with the specified name';

    else IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN Ins_Crs ic ON p.ID =
ic.InsID WHERE ic.CrsID =@CrsID))
        BEGIN
            PRINT 'Access denied. You are not the instructor for this course.';
        END
    else
        begin
            update Question
```

```
            set isDeleted = 1
            where CrsID =  @CrsID
         end
end
GO
GRANT EXECUTE ON  [dbo].[delete_course_questions] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'This stored procedure, delete_course_questions, soft deletes questions associated with a specific course by setting their isDeleted flag to 1. It first verifies if the course exists and checks if the current user is the instructor for the course. If the user is not authorized or the course is not found, appropriate messages are displayed.', 'SCHEMA', N'dbo', 'PROCEDURE', N'delete_course_questions', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Person]
[dbo].[Question]

[dbo].[delete_course_specific_question]

MS_Description

This stored procedure, delete_course_specific_question, soft deletes a specific question (by setting isDeleted to 1) within a given course. It verifies if the course exists, checks if the current user is the instructor for the course, and ensures the question belongs to the specified course. If any condition fails, appropriate messages are displayed.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@CrsName	nvarchar(max)	max
@QuesID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR

SQL Script

```
CREATE   PROCEDURE [dbo].[delete_course_specific_question]
@CrsName NVARCHAR(MAX),
@QuesID Int
As
begin

    declare @CrsID int;
    select @CrsID=id from Course where Name=@CrsName;

    IF @@ROWCOUNT = 0
       PRINT 'No course found with the specified name';

    else IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN Ins_Crs ic ON p.ID =
ic.InsID WHERE ic.CrsID =@CrsID))
        BEGIN
            PRINT 'Access denied. You are not the instructor for this course.';
        END
    else
```

```
        begin
          IF EXISTS (SELECT 1 FROM Question WHERE ID = @QuesID and CrsID= @CrsID )
              begin
                   update Question
                   set isDeleted = 1
                   where ID = @QuesID and CrsID =  @CrsID
               end
           else
            PRINT 'No  Question found with the specified id';
        end
end
GO
GRANT EXECUTE ON  [dbo].[delete_course_specific_question] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'This stored procedure,
delete_course_specific_question, soft deletes a specific question (by setting isDeleted to 1)
within a given course. It verifies if the course exists, checks if the current user is the
instructor for the course, and ensures the question belongs to the specified course. If any
condition fails, appropriate messages are displayed.', 'SCHEMA', N'dbo', 'PROCEDURE',
N'delete_course_specific_question', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Person]
[dbo].[Question]

[dbo].[deleteAllDepartments]

MS_Description

This stored procedure deleteAllDepartments deletes all records from the department table and prints a confirmation message. It is encrypted to prevent viewing its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE proc [dbo].[deleteAllDepartments]
with encryption
as
  begin
      delete from department
      print 'The departments deleted';
  end
GO
GRANT EXECUTE ON  [dbo].[deleteAllDepartments] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'This stored procedure deleteAllDepartments
deletes all records from the department table and prints a confirmation message. It is encrypted
to prevent viewing its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'deleteAllDepartments',
NULL, NULL
GO
```

Uses

[dbo].[Department]

[dbo].[deleteAllExams]

MS_Description

The deleteAllExams stored procedure deletes all records from the exam table and prints a confirmation message. It is encrypted to protect its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   proc [dbo].[deleteAllExams]
with encryption
as
  begin
      delete from exam
      print 'The exams deleted';
  end
GO
GRANT EXECUTE ON  [dbo].[deleteAllExams] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The deleteAllExams stored procedure deletes all records from the exam table and prints a confirmation message. It is encrypted to protect its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'deleteAllExams', NULL, NULL
GO
```

Uses

[dbo].[Exam]

[dbo].[deleteAllInstructorCourseData]

MS_Description

The deleteAllInstructorCourseData stored procedure deletes all records from the Ins_Crs table and prints a confirmation message. It is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE proc [dbo].[deleteAllInstructorCourseData]
with encryption
as
  begin
      delete from Ins_Crs
      print 'The Instructor_Course table deleted';
  end
GO
GRANT EXECUTE ON  [dbo].[deleteAllInstructorCourseData] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The deleteAllInstructorCourseData stored
procedure deletes all records from the Ins_Crs table and prints a confirmation message. It is
encrypted to secure its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'deleteAllInstructorCourse-
Data', NULL, NULL
GO
```

Uses

[dbo].[Ins_Crs]

[dbo].[DeleteInstructorAssignedToCourse]

MS_Description

The DeleteInstructorAssignedToCourse stored procedure removes a specific course assignment for an instructor from the Ins_Crs table. It validates inputs, checks for existence, and handles errors with a TRY...CATCH block. The procedure is encrypted to protect its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@INS_ID	int	4
@CRS_ID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE PROCEDURE [dbo].[DeleteInstructorAssignedToCourse]
    @INS_ID INT = NULL  ,  @CRS_ID INT = NULL
with encryption
AS
BEGIN
  begin try
    IF @INS_ID IS NULL OR @CRS_ID IS NULL
    BEGIN
        print 'Input cannot be empty or NULL.';
    END

  ElSE IF EXISTS (SELECT 1 FROM Ins_Crs WHERE InsID = @INS_ID and CRSid=@CRS_ID)
     begin
       delete from Ins_Crs WHERE  InsID = @INS_ID and CRSid=@CRS_ID
       print 'The course with ID : '+CAST(@CRS_ID AS NVARCHAR(10)) +' for instructor ' +
CAST(@INS_ID AS NVARCHAR(10))+ ' deleted.';
     end
```

```
    ELSE
       PRINT 'The Assignation coures for this instructor does not exist.';
    end try
    BEGIN CATCH
       PRINT 'An error occurred, please try again.';
    END CATCH

END;
GO
GRANT EXECUTE ON  [dbo].[DeleteInstructorAssignedToCourse] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The DeleteInstructorAssignedToCourse stored
procedure removes a specific course assignment for an instructor from the Ins_Crs table. It
validates inputs, checks for existence, and handles errors with a TRY...CATCH block. The
procedure is encrypted to protect its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'Delete-
InstructorAssignedToCourse', NULL, NULL
GO
```

Uses

[dbo].[Ins_Crs]

[dbo].[DeleteInstructorCourses]

MS_Description

The DeleteInstructorCourses stored procedure deletes all course assignments for a given instructor from the Ins_Crs table. It validates input, checks for existence, and handles errors using a TRY...CATCH block. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create PROCEDURE [dbo].[DeleteInstructorCourses]
    @INS_ID INT = NULL
with encryption
AS
BEGIN
  begin try
    IF @INS_ID IS NULL
    BEGIN
        print 'Input cannot be empty or NULL.';
    END

  ElSE IF EXISTS (SELECT * FROM Ins_Crs WHERE InsID = @INS_ID)
     begin
       delete from Ins_Crs WHERE InsID = @INS_ID
       print 'The courses for instructor ' + CAST(@INS_ID AS NVARCHAR(10))+ ' deleted.';
     end

    ELSE
```

```
        PRINT 'The Instructor with ID ' + CAST(@INS_ID AS NVARCHAR(10)) + ' does not exist.';
    end try
    BEGIN CATCH
        PRINT 'An error occurred, please try again.';
    END CATCH

END;
GO
GRANT EXECUTE ON  [dbo].[DeleteInstructorCourses] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The DeleteInstructorCourses stored procedure
deletes all course assignments for a given instructor from the Ins_Crs table. It validates input,
checks for existence, and handles errors using a TRY...CATCH block. The procedure is encrypted to
secure its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'DeleteInstructorCourses', NULL, NULL
GO
```

Uses

[dbo].[Ins_Crs]

[dbo].[deleteIntake]

MS_Description

The deleteIntake stored procedure deletes a specific intake from the Intake table based on its name. It first checks for existence before deletion and prints a success or failure message accordingly.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	varchar(100)	100

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create   proc [dbo].[deleteIntake] @name varchar(100)
as
  IF EXISTS (SELECT 1 FROM Intake WHERE Name = @Name)
  begin
   delete from intake  where name=@name
   print 'deleted successfully.';
  end

  else
  begin
    print 'The intake is not exists'
  end
GO
GRANT EXECUTE ON  [dbo].[deleteIntake] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The deleteIntake stored procedure deletes a
specific intake from the Intake table based on its name. It first checks for existence before
deletion and prints a success or failure message accordingly.', 'SCHEMA', N'dbo', 'PROCEDURE',
N'deleteIntake', NULL, NULL
GO
```

Uses

[dbo].[Intake]

[dbo].[DeleteLoginAndUser]

MS_Description

The DeleteLoginAndUser stored procedure removes a SQL Server login and its corresponding database user if they exist. It first checks for their existence and then executes dynamic SQL to drop them securely.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@EmailToDelete	nvarchar(255)	510

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
--create or alter procedure sp_add_person
--@name nvarchar(max),
--@mail nvarchar(max),
--@nationalID nvarchar(max),
--@address nvarchar(max),
--@gender char(1),
--@salary decimal(18,2),
--@date_of_birth date,
--@phone nvarchar(max),
--@role nvarchar(30),
--@password nvarchar(255),
--@deptID int =null,
--@intakeID int = null,
--@college nvarchar(255) =null,
--@hireDate date =null
--as with encryption
--begin
--    begin try
--        begin transaction
--          if exists(select 1 from Department where ID=@deptID) or @deptID is null
```

```
--         begin
--              INSERT INTO [dbo].[Person]
--               ([Name]
--               ,[Email]
--               ,[NID]
--               ,[Address]
--               ,[Gender]
--               ,[Salary]
--               ,[DOB]
--               ,[Phone]
--               ,[Role]
--               ,[Password]
--               ,[DeptID])
--         VALUES
--               (@name,@mail,@nationalID,@address,@gender,@salary
,@date_of_birth,@phone,@role,@password,@deptID)
--         declare @lastpersonID int = scope_identity();

--     -------------insert into student------------
--     if(@role = 'Student')
--         begin
--         if exists(select 1 from Intake where ID=@intakeID) or @intakeID is null
--          begin
--            insert into Student values (@lastpersonID,@intakeID,@college)
--            print 'person inserted as a student successfully'
--          end
--          else
--            print 'there is no intake with the specified intake id'
--         end
--    else if(@role = 'Instructor')
--        begin
--            insert into Instructor values (@lastpersonID,@hireDate)
--            print 'person inserted as a instructor successfully'
--         end
--         commit transaction;


--     end
--     else
--       print 'the department id you provided is not found '
--    end try ;
--    begin catch
--      print 'the data is invalid'
--      rollback;
--    end catch
--end;
CREATE PROCEDURE [dbo].[DeleteLoginAndUser]
    @EmailToDelete NVARCHAR(255)
AS
BEGIN
    DECLARE @SqlStatement NVARCHAR(MAX);

    -- Check if the login exists
```

```
    IF EXISTS (SELECT 1 FROM sys.server_principals WHERE name = @EmailToDelete)
    BEGIN
        -- Check if the user exists
        IF EXISTS (SELECT 1 FROM sys.database_principals WHERE name = @EmailToDelete)
        BEGIN
            -- Delete the user
            SET @SqlStatement = 'DROP USER ' + QUOTENAME(@EmailToDelete) + ';';
            EXEC sp_executesql @SqlStatement;
        END

        -- Delete the login
        SET @SqlStatement = 'DROP LOGIN ' + QUOTENAME(@EmailToDelete) + ';';
        EXEC sp_executesql @SqlStatement;
    END
END;
GO
GRANT EXECUTE ON  [dbo].[DeleteLoginAndUser] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The DeleteLoginAndUser stored procedure removes
a SQL Server login and its corresponding database user if they exist. It first checks for their
existence and then executes dynamic SQL to drop them securely.', 'SCHEMA', N'dbo', 'PROCEDURE',
N'DeleteLoginAndUser', NULL, NULL
GO
```

Used By

[dbo].[PERSON_Update]
[dbo].[sp_delete_person]

[dbo].[DeleteSpecificDepartment]

MS_Description

The DeleteSpecificDepartment stored procedure deletes a department based on its ID. It checks for null or empty input, verifies existence before deletion, and prints an appropriate message. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@DID	nvarchar(100)	200

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
Create   PROCEDURE [dbo].[DeleteSpecificDepartment]
    @DID NVARCHAR(100)
with encryption
AS
BEGIN
    IF @DID IS NULL OR LTRIM(RTRIM(@DID)) = ''
    BEGIN
        print 'Input cannot be empty or NULL.';
    END

  ElSE IF EXISTS (SELECT 1 FROM Department WHERE ID = @DID)
     begin
       delete from Department WHERE ID = @DID
       print 'The Department ' + @DID +  ' deleted.';
     end

    ELSE
       print 'The department is not exsists.';
```

```
END;
GO
GRANT EXECUTE ON  [dbo].[DeleteSpecificDepartment] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The DeleteSpecificDepartment stored procedure deletes a department based on its ID. It checks for null or empty input, verifies existence before deletion, and prints an appropriate message. The procedure is encrypted to secure its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'DeleteSpecificDepartment', NULL, NULL
GO
```

Uses

[dbo].[Department]

[dbo].[DeleteSpecificExam]

MS_Description

The DeleteSpecificExam stored procedure deletes an exam based on its ID. It verifies input, checks if the caller is an authorized instructor, and ensures the exam exists before deletion. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR

SQL Script

```
create   PROCEDURE [dbo].[DeleteSpecificExam]
    @EX_ID int
with encryption
AS
BEGIN
    declare @crsid int
    select @crsid=CrsID from exam where id=@EX_ID
    IF @EX_ID IS NULL OR LTRIM(RTRIM(@EX_ID)) = ''
    BEGIN
        print 'Input cannot be empty or NULL.';
    END
    ELSE IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN Ins_Crs ic ON p.ID =
ic.InsID WHERE ic.CrsID =@CrsID ))
        BEGIN
            PRINT 'Access denied. You are not the instructor !.';
        END
   ELSE
   begin
      IF EXISTS (SELECT 1 FROM Exam WHERE ID = @EX_ID)
```

```
        begin
        delete from Exam WHERE ID = @EX_ID
        print 'The EXAM ' + @EX_ID +  ' deleted.';
        end

      ELSE
        print 'The Exam is not exsists.';
     end

END;
GO
GRANT EXECUTE ON  [dbo].[DeleteSpecificExam] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The DeleteSpecificExam stored procedure deletes
an exam based on its ID. It verifies input, checks if the caller is an authorized instructor, and
ensures the exam exists before deletion. The procedure is encrypted to secure its definition.',
'SCHEMA', N'dbo', 'PROCEDURE', N'DeleteSpecificExam', NULL, NULL
GO
```

Uses

[dbo].[Exam]
[dbo].[Ins_Crs]
[dbo].[Person]

[dbo].[DeleteStudentAndAssociatedCourses]

MS_Description

The DeleteStudentAndAssociatedCourses stored procedure deletes the courses associated with a specific student based on their ID. It checks if the student exists, handles errors using TRY...CATCH, and prints appropriate messages. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   proc  [dbo].[DeleteStudentAndAssociatedCourses]
  @ST_ID INT=NULL
WITH ENCRYPTION
AS
BEGIN
    BEGIN TRY
        IF @ST_ID is null
           print 'STUDENT ID should not be null OR EMPTY'

        ELSE IF EXISTS (SELECT * FROM Std_Crs WHERE StdID = @ST_ID)
           BEGIN
             DELETE FROM Std_Crs WHERE StdID=@ST_ID
             PRINT 'COURSES FOR STUDENT '+CAST(@ST_ID AS NVARCHAR(10))+' DELETED'
           END
       else
           begin
             print 'STUDENT id: '+CAST(@ST_ID AS NVARCHAR(10))+ ' not fount'
            end
```

```
    END TRY
    BEGIN CATCH
       PRINT 'ERROR OCCURED.'
    END CATCH
END
GO
GRANT EXECUTE ON  [dbo].[DeleteStudentAndAssociatedCourses] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The DeleteStudentAndAssociatedCourses stored
procedure deletes the courses associated with a specific student based on their ID. It checks if
the student exists, handles errors using TRY...CATCH, and prints appropriate messages. The
procedure is encrypted to secure its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'Delete-
StudentAndAssociatedCourses', NULL, NULL
GO
```

Uses

[dbo].[Std_Crs]

[dbo].[DeleteStudentCoureTable]

MS_Description

The DeleteStudentCoureTable stored procedure deletes all records from the Std_Crs table. It uses a TRY...CATCH block to handle potential errors and prints a success or error message accordingly. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
Create PROC [dbo].[DeleteStudentCoureTable]
WITH ENCRYPTION
AS
BEGIN
    BEGIN TRY
           BEGIN
             DELETE FROM Std_Crs
             PRINT 'STUDENT_COURSE DATA DELETED'
           END

    END TRY
    BEGIN CATCH
       PRINT 'ERROR OCCURED.'
    END CATCH
END
GO
GRANT EXECUTE ON  [dbo].[DeleteStudentCoureTable] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The DeleteStudentCoureTable stored procedure
deletes all records from the Std_Crs table. It uses a TRY...CATCH block to handle potential
errors and prints a success or error message accordingly. The procedure is encrypted to secure
its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'DeleteStudentCoureTable', NULL, NULL
GO
```

Uses

[dbo].[Std_Crs]

[dbo].[DropCourseAndEnrolledStudents]

MS_Description

The DropCourseAndEnrolledStudents stored procedure deletes a course and its associated enrolled students based on the provided course ID. It checks for null input, verifies the existence of the course, and handles errors using a TRY...CATCH block. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE proc  [dbo].[DropCourseAndEnrolledStudents]
  @CRS_ID INT=NULL
WITH ENCRYPTION
AS
BEGIN
    BEGIN TRY
        IF @CRS_ID is null
           print 'COURSE ID should not be null OR EMPTY'

        ELSE IF EXISTS (SELECT * FROM Std_Crs WHERE CrsID = @CRS_ID)
           BEGIN
             DELETE FROM Std_Crs WHERE CrsID=@CRS_ID
             PRINT 'COURSE AND IT IS ENROLLED STUDENT '+CAST(@CRS_ID AS NVARCHAR(10))+' ARE
DELETED'
           END
       else
           begin
             print 'COURSE id: '+CAST(@CRS_ID AS NVARCHAR(10))+ ' not fount'
```

```
            end
    END TRY
    BEGIN CATCH
       PRINT 'ERROR OCCURED.'
    END CATCH
END
GO
GRANT EXECUTE ON  [dbo].[DropCourseAndEnrolledStudents] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The DropCourseAndEnrolledStudents stored
procedure deletes a course and its associated enrolled students based on the provided course ID.
It checks for null input, verifies the existence of the course, and handles errors using a
TRY...CATCH block. The procedure is encrypted to secure its definition.', 'SCHEMA', N'dbo',
'PROCEDURE', N'DropCourseAndEnrolledStudents', NULL, NULL
GO
```

Uses

[dbo].[Std_Crs]

[dbo].[generate_exam]

MS_Description

The generate_exam stored procedure creates a new exam for a specific course. It ensures that the course exists and the user is the instructor, then generates the exam and randomly selects questions from the Question table. It also sets up student participation by inserting relevant data into the Answer_Exam and Student_Exam tables. The procedure uses a TRY...CATCH block for error handling and transactions to ensure data consistency. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@examName	nvarchar(100)	200
@startDate	datetime	8
@crsID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR

SQL Script

```
create    procedure [dbo].[generate_exam]
@examName nvarchar(100),
@startDate datetime,
@crsID int
with encryption
as
begin
begin try
  begin transaction
    IF @crsID IS NULL
            PRINT 'Input Course ID cannot be NULL.';
    ELSE IF NOT EXISTS (SELECT 1 FROM COURSE WHERE ID = @CrsID)
            PRINT 'Input Course ID cannot be found';
   ELSE IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN Ins_Crs ic ON p.ID = ic.Ins-
ID WHERE ic.CrsID =@crsID))
```

```
        BEGIN
            PRINT 'Access denied. You are not the instructor for this course.';
            ROLLBACK TRANSACTION;
        END
   else
    begin
        insert into Exam (Name,StartDate,CrsID) values(@examName,@startDate,@crsID)
        declare @examID int=scope_identity();
        create table #selectedRandomQuestionIDS
        (
        QID int
        );
        insert into #selectedRandomQuestionIDS
        SELECT TOP 10 id
                 FROM Question
                 WHERE CrsID = @crsID and isDeleted=0
                 ORDER BY NEWID();
       insert into Answer_Exam (ExamID,QID,StdID)
       select @examID ,QID,sc.StdID from #selectedRandomQuestionIDS cross join Std_Crs SC where
sc.CrsID=@crsID;
       INSERT INTO Student_Exam (StdID, ExamID, Grade)
        SELECT
            StdID AS StdID,
            @examID AS ExamID,
            0 AS Grade
            FROM std_crs where CrsID=@crsID;
       print 'exam is generated successefully'
     commit transaction;
 end
end try
begin catch
 print 'error occured in exam generation '
 rollback;
end catch
end
GO
GRANT EXECUTE ON  [dbo].[generate_exam] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The generate_exam stored procedure creates a new
exam for a specific course. It ensures that the course exists and the user is the instructor,
then generates the exam and randomly selects questions from the Question table. It also sets up
student participation by inserting relevant data into the Answer_Exam and Student_Exam tables.
The procedure uses a TRY...CATCH block for error handling and transactions to ensure data
consistency. The procedure is encrypted to secure its definition.', 'SCHEMA', N'dbo',
'PROCEDURE', N'generate_exam', NULL, NULL
GO
```

Uses

[dbo].[Answer_Exam]
[dbo].[Course]
[dbo].[Exam]
[dbo].[Ins_Crs]

[dbo].[Person]
[dbo].[Question]
[dbo].[Std_Crs]
[dbo].[Student_Exam]

[dbo].[GetCoursesByInstructor]

MS_Description

The GetCoursesByInstructor stored procedure retrieves the courses taught by a specific instructor based on their ID. It checks for null or invalid input, verifies that the instructor exists, and returns details about the instructor and their courses. The procedure uses a TRY...CATCH block to handle potential errors and is encrypted to protect its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE PROCEDURE [dbo].[GetCoursesByInstructor]

    @INS_ID INT = NULL
WITH ENCRYPTION
AS
BEGIN
  begin try
    IF @INS_ID IS NULL
    BEGIN
        print 'Instructor id cannot be empty or NULL.';
    END

    else IF EXISTS (SELECT 1 FROM Ins_Crs WHERE InsID = @INS_ID)
     begin
       select InsID as [instructorID],p.Name as[Instructor_Name] ,  c.Name as [Course_Name]  ,
ic.CrsID AS [Course_ID]
       from Ins_Crs ic join person p
```

```
        on p.ID=ic.InsID
        join Course c on  c.ID=ic.CrsID
        where InsID=@INS_ID
      end

    ELSE
    begin
      PRINT 'The Instructor with ID ' + CAST(@INS_ID AS NVARCHAR(10)) + ' does not exist.';
    end
  end try
    BEGIN CATCH
        PRINT 'An error occurred, please try again.';
    END CATCH
end;

--call
GO
GRANT EXECUTE ON  [dbo].[GetCoursesByInstructor] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[GetCoursesByInstructor] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The GetCoursesByInstructor stored procedure
retrieves the courses taught by a specific instructor based on their ID. It checks for null or
invalid input, verifies that the instructor exists, and returns details about the instructor and
their courses. The procedure uses a TRY...CATCH block to handle potential errors and is encrypted
to protect its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'GetCoursesByInstructor', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Person]

[dbo].[InsertDepartment]

MS_Description

The InsertDepartment stored procedure inserts a new department into the Department table if the department name does not already exist. It checks for empty or null input and handles errors using a TRY...CATCH block. The procedure prints appropriate success or error messages and is encrypted to protect its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@Name	nvarchar(100)	200
@Describe	nvarchar(255)	510

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
Create PROCEDURE [dbo].[InsertDepartment]
    @Name NVARCHAR(100),
    @Describe NVARCHAR(255)
with encryption
AS
BEGIN
    IF @Name IS NULL OR LTRIM(RTRIM(@Name)) = ''
    BEGIN
        print 'Input cannot be empty or NULL.';
    END

  ElSE IF EXISTS (SELECT 1 FROM Department WHERE Name = @Name)
     begin
       print 'The Department ' + @Name +  ' Is already exists.';
     end
```

```
    ELSE
    BEGIN try
       INSERT INTO Department(Name, Describe)
       VALUES (@Name, @Describe);
       print 'Data inserted successfully.';
    END try
    begin catch
         select 'invalid input'
    end catch

END;
GO
GRANT EXECUTE ON  [dbo].[InsertDepartment] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The InsertDepartment stored procedure inserts a
new department into the Department table if the department name does not already exist. It checks
for empty or null input and handles errors using a TRY...CATCH block. The procedure prints
appropriate success or error messages and is encrypted to protect its definition.', 'SCHEMA',
N'dbo', 'PROCEDURE', N'InsertDepartment', NULL, NULL
GO
```

Uses

[dbo].[Department]

[dbo].[InsertInstructorCourse]

MS_Description

The InsertInstructorCourse stored procedure assigns an instructor to a course by inserting a record into the Ins_Crs table. It validates the inputs, ensures that both the instructor and course exist, and checks if the assignment already exists. The procedure uses a TRY...CATCH block for error handling and prints appropriate success or error messages. It is encrypted to protect its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@INS_ID	int	4
@CRS_ID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE PROCEDURE [dbo].[InsertInstructorCourse]
     @INS_ID INT = NULL  ,  @CRS_ID INT = NULL
WITH ENCRYPTION
AS
BEGIN
  BEGIN TRY
     IF @INS_ID IS NULL
       BEGIN
        print 'Input INS_ID cannot be empty or NULL.';
       END

     ELSE IF @CRS_ID IS NULL
            PRINT 'Input Course ID cannot be NULL.';

     ELSE IF NOT EXISTS (SELECT * FROM COURSE WHERE ID = @CRS_ID)
            PRINT 'The Course with ID: ' + CAST(@CRS_ID AS NVARCHAR(10)) + ' is not found.';
```

```
     ELSE IF NOT EXISTS (SELECT * FROM Instructor WHERE InsID = @INS_ID)
            PRINT 'The INSTRUCTOR with ID: ' + CAST(@INS_ID AS NVARCHAR(10)) + ' is not found.';
    ELSE IF EXISTS (SELECT * FROM Ins_Crs WHERE InsID = @INS_ID and CrsID=@CRS_ID)
            PRINT 'The Assigned Between ' + CAST(@INS_ID AS NVARCHAR(10))+' and'+CAST(@CRS_ID AS
NVARCHAR(10)) + ' is already exists.';
     ELSE
        BEGIN
            INSERT INTO Ins_Crs( InsID, CrsID)
            VALUES (@INS_ID,@CRS_ID);
            PRINT 'Data inserted successfully.';
        END
   END TRY
   BEGIN CATCH
            PRINT 'An error occurred.';
   END CATCH
END;
GO
GRANT EXECUTE ON  [dbo].[InsertInstructorCourse] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The InsertInstructorCourse stored procedure
assigns an instructor to a course by inserting a record into the Ins_Crs table. It validates the
inputs, ensures that both the instructor and course exist, and checks if the assignment already
exists. The procedure uses a TRY...CATCH block for error handling and prints appropriate success
or error messages. It is encrypted to protect its definition.', 'SCHEMA', N'dbo', 'PROCEDURE',
N'InsertInstructorCourse', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Instructor]
INSTRUCTOR

[dbo].[InsertIntake]

MS_Description

The InsertIntake stored procedure inserts a new intake record into the Intake table. It validates that the name is not empty, ensures the start date is not in the past, and checks if the user is an admin before proceeding. It uses a TRY...CATCH block for error handling and prints appropriate messages for success or failure.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Name	nvarchar(100)	200
@StartDate	date	3

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[InsertIntake]
    @Name NVARCHAR(100),
    @StartDate DATE
AS
BEGIN
 BEGIN TRY
    IF @Name IS NULL OR LTRIM(RTRIM(@Name)) = ''
    BEGIN
        print 'Name cannot be empty or NULL.';
    END

    ELSE IF @StartDate < GETDATE()  --2023 > 2022
    BEGIN
        print 'StartDate cannot be in the past.';
    END
   ELSE IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p WHERE  p.Role='admin' ))
        BEGIN
            PRINT 'Access denied. You are not the ADMIN !.';
```

```
        END
    ELSE IF (SUSER_SNAME() in (SELECT  p.Email FROM Person p WHERE  p.Role='admin' ))
    BEGIN
    INSERT INTO Intake (Name, StartDate)
    VALUES (@Name, @StartDate);

    print 'Data inserted successfully.';
    END
    END try
    begin catch
         select 'invalid input'
    end catch


END;
GO
GRANT EXECUTE ON  [dbo].[InsertIntake] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The InsertIntake stored procedure inserts a new
intake record into the Intake table. It validates that the name is not empty, ensures the start
date is not in the past, and checks if the user is an admin before proceeding. It uses a
TRY...CATCH block for error handling and prints appropriate messages for success or failure.',
'SCHEMA', N'dbo', 'PROCEDURE', N'InsertIntake', NULL, NULL
GO
```

Uses

[dbo].[Intake]
[dbo].[Person]

[dbo].[InsertStudentCourse]

MS_Description

The InsertStudentCourse stored procedure assigns a student to a course by inserting a record into the Std_Crs table. It validates the student and course IDs, checks if the assignment already exists, and ensures that both the student and course are valid. The procedure uses a TRY...CATCH block for error handling and prints appropriate messages for success or error. It is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@St_ID	int	4
@CRS_ID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE  PROCEDURE [dbo].[InsertStudentCourse]
     @St_ID INT = NULL  ,  @CRS_ID INT = NULL
WITH ENCRYPTION
AS
BEGIN
  BEGIN TRY
     IF @St_ID IS NULL
       BEGIN
        print 'Input Student_ID cannot be empty or NULL.';
       END

     ELSE IF @CRS_ID IS NULL
            PRINT 'Input Course ID cannot be NULL.';
     ELSE IF NOT EXISTS (SELECT * FROM Student WHERE StdID = @St_ID)
            PRINT 'The Student with ID: ' + CAST(@St_ID AS NVARCHAR(10)) + ' is not found.';
     ELSE IF NOT EXISTS (SELECT * FROM COURSE WHERE ID = @CRS_ID)
```

```
            PRINT 'The Course with ID: ' + CAST(@CRS_ID AS NVARCHAR(10)) + ' is not found.';
    ELSE IF EXISTS (SELECT * FROM Std_Crs WHERE StdID = @St_ID and CrsID=@CRS_ID)
            PRINT 'The Assigned Between ' + CAST(@St_ID AS NVARCHAR(10))+' and'+CAST(@CRS_ID AS
NVARCHAR(10)) + ' is already exists.';
    ELSE
        BEGIN
            INSERT INTO Std_Crs( StdID, CrsID)
            VALUES (@St_ID,@CRS_ID);
            PRINT 'Data inserted successfully.';
        END
   END TRY
   BEGIN CATCH
            PRINT 'An error occurred.';
   END CATCH
END;
GO
GRANT EXECUTE ON  [dbo].[InsertStudentCourse] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The InsertStudentCourse stored procedure assigns
a student to a course by inserting a record into the Std_Crs table. It validates the student and
course IDs, checks if the assignment already exists, and ensures that both the student and course
are valid. The procedure uses a TRY...CATCH block for error handling and prints appropriate
messages for success or error. It is encrypted to secure its definition.', 'SCHEMA', N'dbo',
'PROCEDURE', N'InsertStudentCourse', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Std_Crs]
[dbo].[Student]
STUDENT

[dbo].[ListStudentsForCourse]

MS_Description

The ListStudentsForCourse stored procedure retrieves a list of students enrolled in a specific course based on the provided course ID. It checks if the course exists and returns student details, including their ID, name, and the course name. If the course doesn't exist, it prints a message indicating so. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
create PROC [dbo].[ListStudentsForCourse]
    @Cr_id int=null
WITH ENCRYPTION
AS
 begin
 if @Cr_id is null
        print 'Course ID should not be null OR EMPTY'
 ELSE IF EXISTS (SELECT * FROM Std_Crs WHERE CrsID = @Cr_id)
       BEGIN
          SELECT s.StdID as [student_id] ,p.Name as [student_name]  ,c.Name as [course_name]
          FROM Std_Crs s join person p
          on s.StdID=p.id
          join Course c on CrsID=c.ID
         where CrsID = @Cr_id
        END
```

```
 else
   begin
     print 'course id: '+CAST(@Cr_id AS NVARCHAR(10))+ ' not fount'
   end
end
GO
GRANT EXECUTE ON  [dbo].[ListStudentsForCourse] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[ListStudentsForCourse] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ListStudentsForCourse stored procedure
retrieves a list of students enrolled in a specific course based on the provided course ID. It
checks if the course exists and returns student details, including their ID, name, and the course
name. If the course doesn''t exist, it prints a message indicating so. The procedure is encrypted
to secure its definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'ListStudentsForCourse', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Person]
[dbo].[Std_Crs]

[dbo].[ModifyCourseInstructor]

MS_Description

The ModifyCourseInstructor stored procedure allows you to update the instructor assigned to a specific course. It checks if the old and new instructor IDs, as well as the course ID, are valid. If the old instructor is assigned to the course, it will update the record to reflect the new instructor unless the new assignment already exists. The procedure ensures that all required conditions are met and handles errors with a TRY...CATCH block. It prints messages based on the outcome of the operation, providing feedback to the user.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@oldINS_ID	int	4
@CRS_ID	int	4
@newIns_ID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create PROCEDURE [dbo].[ModifyCourseInstructor]
    @oldINS_ID INT = NULL  ,  @CRS_ID INT = NULL   ,@newIns_ID INT = NULL
with encryption
AS
BEGIN
  begin try
     IF @oldINS_ID IS NULL or @newins_ID is null
       BEGIN
        print 'Input INSTRUCTOR_ID cannot be empty or NULL.';
       END

     ELSE IF @CRS_ID IS NULL
            PRINT 'Input Course ID cannot be NULL.';
     ELSE IF NOT EXISTS (SELECT * FROM Instructor WHERE InsID =@oldINS_ID)
```

```
            PRINT 'The old instructor with ID: ' + CAST(@oldINS_ID AS NVARCHAR(10)) + ' is not
found.';
     ELSE IF NOT EXISTS (SELECT * FROM COURSE WHERE ID = @CRS_ID)
            PRINT 'The Course with ID: ' + CAST(@CRS_ID AS NVARCHAR(10)) + ' is not found.';
     ELSE IF NOT EXISTS (SELECT * FROM Instructor WHERE InsID = @newIns_ID)
            PRINT 'The new Instructor with ID: ' + CAST(@newIns_ID AS NVARCHAR(10)) + ' is not
found.';

     ElSE IF EXISTS (SELECT 1 FROM Ins_Crs WHERE InsID = @oldINS_ID and CRSid=@CRS_ID)  --ins_id
, old
       begin
         IF not EXISTS (SELECT 1 FROM Ins_Crs WHERE InsID = @newIns_ID and CRSid=@CRS_ID)
             begin
                update Ins_Crs set InsID=@newIns_ID
                where InsID=@oldINS_ID and CrsID=@CRS_ID
              end
         else
         begin
             print 'the assgination with instructor id: '+CAST(@newIns_ID AS NVARCHAR(10))+' and
course id: '+CAST(@CRS_ID AS NVARCHAR(10))+ ' already exists'
         end
       end
    else
     begin
          print 'the assgination with instructor id: '+CAST(@oldINS_ID AS NVARCHAR(10))+' and
course id: '+CAST(@CRS_ID AS NVARCHAR(10))+ ' is not exists'
     end
    end try
    BEGIN CATCH
       PRINT 'An error occurred, please try again.';
    END CATCH

END;
GO
GRANT EXECUTE ON  [dbo].[ModifyCourseInstructor] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ModifyCourseInstructor stored procedure
allows you to update the instructor assigned to a specific course. It checks if the old and new
instructor IDs, as well as the course ID, are valid. If the old instructor is assigned to the
course, it will update the record to reflect the new instructor unless the new assignment already
exists. The procedure ensures that all required conditions are met and handles errors with a
TRY...CATCH block. It prints messages based on the outcome of the operation, providing feedback
to the user.', 'SCHEMA', N'dbo', 'PROCEDURE', N'ModifyCourseInstructor', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Instructor]
INSTRUCTOR

[dbo].[ModifyInstructorCourse]

MS_Description

This stored procedure updates the course assignment for an instructor. It validates input parameters for instructor and course existence, checks if the assignment exists, and performs the update. If any validation fails, it prints an appropriate message.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@INS_ID	int	4
@Old_CRS_ID	int	4
@new_CRS_ID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE PROCEDURE [dbo].[ModifyInstructorCourse]
    @INS_ID INT = NULL  ,  @Old_CRS_ID INT = NULL   ,@new_CRS_ID INT = NULL
with encryption
AS
BEGIN
  begin try
     IF @INS_ID IS NULL
       BEGIN
        print 'Input INSTRUCTOR_ID cannot be empty or NULL.';
       END

     ELSE IF @Old_CRS_ID IS NULL OR @new_CRS_ID IS NULL
            PRINT 'Input Course ID cannot be NULL.';
     ELSE IF NOT EXISTS (SELECT * FROM Instructor WHERE InsID =@INS_ID)
            PRINT 'The instructor with ID: ' + CAST(@INS_ID AS NVARCHAR(10)) + ' is not found.';
     ELSE IF NOT EXISTS (SELECT * FROM COURSE WHERE ID = @Old_CRS_ID)
```

```
            PRINT 'The old Course with ID: ' + CAST(@Old_CRS_ID AS NVARCHAR(10)) + ' is not
found.';
     ELSE IF NOT EXISTS (SELECT * FROM COURSE WHERE ID = @new_CRS_ID)
            PRINT 'The new Course with ID: ' + CAST(@new_CRS_ID AS NVARCHAR(10)) + ' is not
found.';

     ElSE IF EXISTS (SELECT 1 FROM Ins_Crs WHERE InsID = @INS_ID and CRSid=@Old_CRS_ID)  --ins_id
, old
       begin
         IF not EXISTS (SELECT 1 FROM Ins_Crs WHERE InsID = @INS_ID and CRSid=@new_CRS_ID)
             begin
                update Ins_Crs set CrsID=@new_CRS_ID
                where InsID=@INS_ID and CrsID=@Old_CRS_ID
              end
         else
         begin
             print 'the assgination with instructor id: '+CAST(@INS_ID AS NVARCHAR(10))+' and
course id: '+CAST(@new_CRS_ID AS NVARCHAR(10))+ ' already exists'
         end
       end
    else
     begin
          print 'the assgination with instructor id: '+CAST(@INS_ID AS NVARCHAR(10))+' and course
id: '+CAST(@Old_CRS_ID AS NVARCHAR(10))+ ' is not exists'
     end
    end try
    BEGIN CATCH
       PRINT 'An error occurred, please try again.';
    END CATCH

END;
GO
GRANT EXECUTE ON  [dbo].[ModifyInstructorCourse] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'This stored procedure updates the course
assignment for an instructor. It validates input parameters for instructor and course existence,
checks if the assignment exists, and performs the update. If any validation fails, it prints an
appropriate message.', 'SCHEMA', N'dbo', 'PROCEDURE', N'ModifyInstructorCourse', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Instructor]
INSTRUCTOR

[dbo].[PERSON_Update]

MS_Description

The PERSON_Update stored procedure updates a person's email in the Person table while ensuring the email is unique and the ID exists. It deletes the old login, updates the email, and then creates a new login and user with the updated email, preserving the person's role and password.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@ID	int	4
@Email	nvarchar(450)	900

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[PERSON_Update]
@ID int,
@Email nvarchar(450)

WITH ENCRYPTION
AS
BEGIN   -- Update Process for Student Table

    declare @IDException bit = 0;
    declare @EmailException bit = 0;

    if exists(select * from Person P where P.Email = @Email and P.ID!= @ID)
        set @EmailException = 1; -- If the Email already exists and it is not the same as student
being updated then, set the exception flag to 1, indicating that the inserted Email is not unique

    if exists(select * from Person P where P.ID= @ID)
        set @IDException = 1; -- If the Student exists, set the exception flag to 1
```

```
    if @IDException = 0
        print 'An error has occured, The Student ID you entered does not exist in Student Table'
    else if @EmailException = 1
        print 'An error has occured, the Email you entered already exists , enter a unique Email'
    else
        BEGIN
            DECLARE @DefaultDatabase NVARCHAR(255) = 'YEARO_EXAM_SYSTEM';
            DECLARE @SqlStatement NVARCHAR(MAX);
            DECLARE @oldEmail nvarchar(450);
            SELECT @oldEmail = p.Email FROM person p WHERE p.ID = @ID;

            EXEC DeleteLoginAndUser @EmailToDelete = @oldEmail

            UPDATE Person  set  email = @Email  where ID = @ID;
            declare @per_role NVARCHAR(MAX);
            select @per_role = p.Role  from person p where p.ID=@ID
            declare @per_password NVARCHAR(MAX);
            select @per_password = p.Password  from person p where p.ID=@ID

            -- Construct the dynamic SQL statement
            SET @SqlStatement = 'CREATE LOGIN ' + QUOTENAME(@Email) + ' WITH PASSWORD = ''' +
@per_password + ''', DEFAULT_DATABASE = ' + QUOTENAME(@DefaultDatabase) + ';';
            EXEC sp_executesql @SqlStatement;

            -- Create user and add to role
             SET @SqlStatement = 'CREATE USER ' + QUOTENAME(@Email) + ' FOR LOGIN ' +
QUOTENAME(@Email) + ';';
            EXEC sp_executesql @SqlStatement;

            SET @SqlStatement = 'ALTER ROLE '+ QUOTENAME(@per_role)+' ADD MEMBER ' +
QUOTENAME(@Email) + ';';
            EXEC sp_executesql @SqlStatement;
        END
END;
GO
GRANT EXECUTE ON  [dbo].[PERSON_Update] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The PERSON_Update stored procedure updates a
person''s email in the Person table while ensuring the email is unique and the ID exists. It
deletes the old login, updates the email, and then creates a new login and user with the updated
email, preserving the person''s role and password.', 'SCHEMA', N'dbo', 'PROCEDURE', N'PERSON_-
Update', NULL, NULL
GO
```

Uses

[dbo].[Person]
[dbo].[DeleteLoginAndUser]

[dbo].[pro_insert_question_choice]

MS_Description

The pro_insert_question_choice stored procedure inserts a question with choices into the Question and Choice tables based on the provided course name. It ensures that the user is the instructor for the course, validates the choices for MCQs, and handles both MCQ and true/false question types. If all conditions are met, it commits the transaction; otherwise, it rolls back.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Body	nvarchar(max)	max
@Type	nvarchar(10)	20
@Degree	int	4
@CorrectChoice	int	4
@CrsName	nvarchar(max)	max
@Choice1	nvarchar(max)	max
@Choice2	nvarchar(max)	max
@Choice3	nvarchar(max)	max
@Choice4	nvarchar(max)	max

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR

SQL Script

```
CREATE   PROCEDURE [dbo].[pro_insert_question_choice]
    @Body NVARCHAR(MAX),
    @Type NVARCHAR(10) ,
    @Degree INT ,
    @CorrectChoice INT,
    @CrsName NVARCHAR(MAX),
    @Choice1 NVARCHAR(MAX) = NULL,
    @Choice2 NVARCHAR(MAX) = NULL,
    @Choice3 NVARCHAR(MAX) = NULL,
```

```
    @Choice4 NVARCHAR(MAX) = NULL

AS
BEGIN
  begin try
    begin transaction;

    declare @CrsID int;
    select @CrsID=id from Course where Name=@CrsName;

    IF @@ROWCOUNT = 0
    PRINT 'No course found with the specified name';

   ELSE IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN Ins_Crs ic ON p.ID = ic.Ins-
ID WHERE ic.CrsID =@CrsID))
        BEGIN
            PRINT 'Access denied. You are not the instructor for this course.';
            ROLLBACK TRANSACTION;
        END
    ELSE
    begin
    INSERT INTO Question (Body, [Type], Degree, CorrectChoice, CrsID)
    VALUES (@Body, @Type, @Degree, @CorrectChoice, @CrsID);

    DECLARE @question_id INT;
    SELECT @question_id = SCOPE_IDENTITY();

    if( @Type='mcq')
    begin
         if( @Choice1 IS NOT NULL AND @Choice2 IS NOT NULL AND
              @Choice3 IS NOT NULL AND @Choice4 IS NOT NULL )
             begin
                  INSERT INTO Choice(QID,Choice,body)
                  VALUES (@question_id,1, @Choice1);
                  INSERT INTO Choice(QID,Choice,body)
                  VALUES (@question_id,2, @Choice2);
                  INSERT INTO Choice(QID,Choice,body)
                  VALUES (@question_id,3, @Choice3);
                  INSERT INTO Choice(QID,Choice,body)
                  VALUES (@question_id,4, @Choice4);
            commit transaction;
            end
          ELSE
            BEGIN
                RAISERROR('All choices must be provided for MCQ questions.', 16, 1);
            END

     end
    else
        begin
                INSERT INTO Choice(QID,Choice,body)
                VALUES (@question_id,1, 'T');
                INSERT INTO Choice(QID,Choice,body)
```

```
                VALUES (@question_id,2, 'F');
                commit transaction;
        end
    end
end try
begin catch
rollback;
print 'error occured when insert qustion'
end catch
END;
GO
GRANT EXECUTE ON  [dbo].[pro_insert_question_choice] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The pro_insert_question_choice stored procedure
inserts a question with choices into the Question and Choice tables based on the provided course
name. It ensures that the user is the instructor for the course, validates the choices for MCQs,
and handles both MCQ and true/false question types. If all conditions are met, it commits the
transaction; otherwise, it rolls back.', 'SCHEMA', N'dbo', 'PROCEDURE',
N'pro_insert_question_choice', NULL, NULL
GO
```

Uses

[dbo].[Choice]
[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Person]
[dbo].[Question]

[dbo].[sp_add_person]

MS_Description

The sp_add_person stored procedure inserts a new person into the Person table, handling different roles (Student, Instructor, Admin). It checks if the provided department and intake IDs exist, creates the appropriate login and user, and adds them to relevant roles. The procedure commits the transaction if successful, or rolls it back in case of an error.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	nvarchar(max)	max
@mail	nvarchar(max)	max
@nationalID	nvarchar(max)	max
@address	nvarchar(max)	max
@gender	char	1
@salary	decimal(18,2)	9
@date_of_birth	date	3
@phone	nvarchar(max)	max
@role	nvarchar(30)	60
@password	nvarchar(255)	510
@deptID	int	4
@intakeID	int	4
@college	nvarchar(255)	510
@hireDate	date	3

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_add_person]
```

```
    @name NVARCHAR(MAX),
    @mail NVARCHAR(MAX),
    @nationalID NVARCHAR(MAX),
    @address NVARCHAR(MAX),
    @gender CHAR(1),
    @salary DECIMAL(18, 2),
    @date_of_birth DATE,
    @phone NVARCHAR(MAX),
    @role NVARCHAR(30),
    @password NVARCHAR(255),
    @deptID INT = NULL,
    @intakeID INT = NULL,
    @college NVARCHAR(255) = NULL,
    @hireDate DATE = NULL
WITH ENCRYPTION
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        -- Check if the department exists or is null
        IF EXISTS (SELECT 1 FROM Department WHERE ID = @deptID) OR @deptID IS NULL
        BEGIN
            -- Insert into Person table
            INSERT INTO [dbo].[Person]
                ([Name], [Email], [NID], [Address], [Gender], [Salary], [DOB], [Phone], [Role],
[Password], [DeptID])
            VALUES
                (@name, @mail, @nationalID, @address, @gender, @salary, @date_of_birth, @phone,
@role, @password, @deptID);

            -- Get the last inserted Person ID
            DECLARE @lastpersonID INT = SCOPE_IDENTITY();
            DECLARE @DefaultDatabase NVARCHAR(255) = 'YEARO_EXAM_SYSTEM';
            DECLARE @SqlStatement NVARCHAR(MAX);

            -- Create login and user
            SET @SqlStatement = 'CREATE LOGIN ' + QUOTENAME(@mail) + ' WITH PASSWORD = ''' +
@password + ''', DEFAULT_DATABASE = ' + QUOTENAME(@DefaultDatabase) + ';';
            EXEC sp_executesql @SqlStatement;

            SET @SqlStatement = 'CREATE USER ' + QUOTENAME(@mail) + ' FOR LOGIN ' +
QUOTENAME(@mail) + ';';
            EXEC sp_executesql @SqlStatement;



            -- Handle Student role
            IF (@role = 'Student')
            BEGIN
                IF EXISTS (SELECT 1 FROM Intake WHERE ID = @intakeID) OR @intakeID IS NULL
                BEGIN
                    INSERT INTO Student (StdID, IntakeID, College)
                    VALUES (@lastpersonID, @intakeID, @college);
```

```
                    SET @SqlStatement = 'ALTER ROLE STUDENT ADD MEMBER ' + QUOTENAME(@mail) +
';';
                    EXEC sp_executesql @SqlStatement;

                    PRINT 'Person inserted as a student successfully.';
                END
                ELSE
                BEGIN
                    PRINT 'There is no intake with the specified intake ID.';
                END
            END
            -- Handle Instructor role
            ELSE IF (@role = 'Instructor')
            BEGIN
                INSERT INTO Instructor (InsID, HireDate)
                VALUES (@lastpersonID, @hireDate);

                SET @SqlStatement = 'ALTER ROLE INSTRUCTOR ADD MEMBER ' + QUOTENAME(@mail) + ';';
                EXEC sp_executesql @SqlStatement;

                PRINT 'Person inserted as an instructor successfully.';
            END
            -- Handle Admin role
            ELSE
            BEGIN
                SET @SqlStatement = 'ALTER ROLE ADMIN ADD MEMBER ' + QUOTENAME(@mail) + ';';
                EXEC sp_executesql @SqlStatement;

                PRINT 'Person inserted as an admin successfully.';
            END

            COMMIT TRANSACTION;
        END
        ELSE
        BEGIN
            PRINT 'The department ID you provided is not found.';
        END
    END TRY
    BEGIN CATCH
        -- Handle errors and roll back transaction
        PRINT  ERROR_MESSAGE();
        ROLLBACK TRANSACTION;
    END CATCH
END;
GO
GRANT EXECUTE ON  [dbo].[sp_add_person] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_add_person stored procedure inserts a new
person into the Person table, handling different roles (Student, Instructor, Admin). It checks if
the provided department and intake IDs exist, creates the appropriate login and user, and adds
them to relevant roles. The procedure commits the transaction if successful, or rolls it back in
case of an error.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_add_person', NULL, NULL
GO
```

Uses

[dbo].[Department]
[dbo].[Instructor]
[dbo].[Intake]
[dbo].[Person]
[dbo].[Student]
INSTRUCTOR
STUDENT

[dbo].[sp_AddCourse]

MS_Description

The sp_AddCourse stored procedure adds a new course to the Course table after validating that the course hours are between 3 and 100. If the hours are outside this range, an error message is raised. If valid, the course is inserted into the table.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Name	nvarchar(100)	200
@Hours	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_AddCourse]
    @Name NVARCHAR(100),
    @Hours INT
AS
BEGIN
    IF (@Hours < 3 OR @Hours > 100)
    BEGIN
        RAISERROR ('Course hours must be between 3 and 100.', 16, 1);
        RETURN;
    END
    else
        begin
        INSERT INTO Course (Name, Hours)
        VALUES (@Name, @Hours);
        end
END;
GO
GRANT EXECUTE ON  [dbo].[sp_AddCourse] TO [ADMIN]
```

```
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_AddCourse stored procedure adds a new course to the Course table after validating that the course hours are between 3 and 100. If the hours are outside this range, an error message is raised. If valid, the course is inserted into the table.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_AddCourse', NULL, NULL
GO
```

Uses

[dbo].[Course]

[dbo].[sp_AddTopic]

MS_Description

The sp_AddTopic stored procedure adds a new topic to the Topic table, associating it with a specified course. It first checks if the course exists by name; if not, it prints an error message. If the course is found, the topic is inserted with the corresponding course ID.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@topicName	nvarchar(100)	200
@CrsName	nvarchar(100)	200

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_AddTopic]
    @topicName NVARCHAR(100),
    @CrsName NVARCHAR(100)
AS
BEGIN
    declare @id int;
    select @id=id from Course where Name=@CrsName;
    IF @@ROWCOUNT = 0
        PRINT 'No course found with the specified name';
    else
        INSERT INTO [dbo].[Topic] ([Name], [CrsID])
        VALUES (@topicName, @id);
END;
GO
GRANT EXECUTE ON  [dbo].[sp_AddTopic] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_AddTopic stored procedure adds a new
topic to the Topic table, associating it with a specified course. It first checks if the course
```

```
exists by name; if not, it prints an error message. If the course is found, the topic is inserted
with the corresponding course ID.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_AddTopic', NULL, NULL
GO
```

Uses

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

[dbo].[sp_calc_student_grade]

MS_Description

The sp_calc_student_grade stored procedure calculates a student's grade for a specific exam by comparing the answers with the correct choices. It computes the total grade, percentage, and updates the Student_Exam table with the student's grade. If the data is invalid or an error occurs, the transaction is rolled back.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@stdID	int	4
@examID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	STUDENT

SQL Script

```
create   procedure [dbo].[sp_calc_student_grade]
@stdID int,
@examID int
with encryption
as
begin
   begin try
       begin transaction
       if exists(select 1 from Student_Exam where ExamID=@examID and  StdID=@stdID)
        begin
         declare @exam_total_grade int;
         declare @student_total_grade int;
         declare @student_total_percentage_grade int;
         select @exam_total_grade=sum(degree) from Question Q inner join Answer_Exam AE on
Q.ID=AE.QID
          and ExamID=@examID and StdID=@stdID
         select @student_total_grade=sum(degree) from Question Q inner join Answer_Exam AE on
```

```
Q.ID=AE.QID
          and ExamID=@examID and StdID=@stdID and Answer=CorrectChoice
         update Student_Exam set Grade=@student_total_grade where ExamID=@examID and StdID=@stdID
         set @student_total_percentage_grade=(CAST(@student_total_grade AS DECIMAL(10,2)) /
                                          CAST(@exam_total_grade AS DECIMAL(10,2)))*100;
         --print 'your percentage grade is '+ CAST(@student_total_percentage_grade AS
NVARCHAR(10)) + ' %';
         commit transaction;
        end
       else
       rollback;
         print 'your data is not valid '
   end try
   begin catch
       rollback;
      PRINT 'Error occurred while calculating the grade.';
   end catch
end
GO
GRANT EXECUTE ON  [dbo].[sp_calc_student_grade] TO [STUDENT]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_calc_student_grade stored procedure
calculates a student''s grade for a specific exam by comparing the answers with the correct
choices. It computes the total grade, percentage, and updates the Student_Exam table with the
student''s grade. If the data is invalid or an error occurs, the transaction is rolled back.',
'SCHEMA', N'dbo', 'PROCEDURE', N'sp_calc_student_grade', NULL, NULL
GO
```

Uses

[dbo].[Answer_Exam]
[dbo].[Question]
[dbo].[Student_Exam]

Used By

[dbo].[sp_submit_exam_answer]

[dbo].[sp_ChangeTopicCourse]

MS_Description

The sp_ChangeTopicCourse stored procedure changes the course assignment of a specific topic. It checks if the topic and both courses exist, then updates the topic's course ID to the new course. If any of the conditions fail, it prints an appropriate error message.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@TopicName	nvarchar(100)	200
@oldCourseName	nvarchar(100)	200
@newCourseName	nvarchar(100)	200

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_ChangeTopicCourse]
    @TopicName NVARCHAR(100),
    @oldCourseName NVARCHAR(100),
    @newCourseName NVARCHAR(100)
AS
BEGIN
    IF EXISTS (SELECT 1 FROM Topic WHERE name = @TopicName)and
        EXISTS (SELECT 1 FROM course WHERE name = @oldCourseName ) and
        EXISTS (SELECT 1 FROM course WHERE name = @newCourseName )
        begin
            declare @oldCourseId int;
            declare @newCourseId int;
            select @oldCourseId=id from Course where Name=@oldCourseName;
            select @newCourseId=id from Course where Name=@newCourseName;
            UPDATE [dbo].[Topic]
            SET [CrsID] = @newCourseId
```

```
            WHERE [CrsID] = @oldCourseId and Name=@TopicName;
        end
    else if not EXISTS (SELECT 1 FROM Topic WHERE name = @TopicName)
     print 'No topic found with the specified name'
    else if not EXISTS (SELECT 1 FROM course WHERE name = @oldCourseName )
     print 'the course  you want to move the topic from does not exist '
    else if not EXISTS (SELECT 1 FROM course WHERE name = @newCourseName )
     print 'the course  you want to move the topic to does not exist '
END;
GO
GRANT EXECUTE ON  [dbo].[sp_ChangeTopicCourse] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_ChangeTopicCourse stored procedure
changes the course assignment of a specific topic. It checks if the topic and both courses exist,
then updates the topic''s course ID to the new course. If any of the conditions fail, it prints
an appropriate error message.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_ChangeTopicCourse', NULL,
NULL
GO
```

Uses

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

[dbo].[sp_delete_person]

MS_Description

The sp_delete_person stored procedure deletes a person from the Person table based on the provided person ID. It checks if the ID is valid and exists, then deletes the corresponding record and removes the associated login using the DeleteLoginAndUser procedure. If the ID is null or the person does not exist, it prints an appropriate message.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_delete_person]
@personID int = null
WITH ENCRYPTION
AS
BEGIN

 IF @personID is null
          print 'person ID should not be null OR EMPTY'

 ELSE IF EXISTS (SELECT * FROM Person WHERE id = @personID)
 begin
     declare @email_for_id nvarchar(255)
     select @email_for_id =p.Email from person p where p.ID=@personID
    delete
    FROM
        Person
    where
        ID=@personID;
```

```
        exec DeleteLoginAndUser @email_for_id;
 end
 else
   print 'the person you want to delete does not exist in the person table'
END;
GO
GRANT EXECUTE ON  [dbo].[sp_delete_person] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_delete_person stored procedure deletes a
person from the Person table based on the provided person ID. It checks if the ID is valid and
exists, then deletes the corresponding record and removes the associated login using the Delete-
LoginAndUser procedure. If the ID is null or the person does not exist, it prints an appropriate
message.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_delete_person', NULL, NULL
GO
```

Uses

[dbo].[Person]
[dbo].[DeleteLoginAndUser]

[dbo].[sp_DeleteCourse]

MS_Description

The sp_DeleteCourse stored procedure deletes a course and its related data. It first checks if the course exists by name. If found, it deletes associated topics, question choices, and related questions before deleting the course itself. If the course does not exist, it prints an error message.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	nvarchar(100)	200

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_DeleteCourse]
    @name NVARCHAR(100)
AS
BEGIN
    declare @id int;
    select @id=id from Course where Name=@name;
    IF @@ROWCOUNT = 0
        PRINT 'No course found with the specified name';
    else
            IF EXISTS (SELECT 1 FROM Topic WHERE CrsID = @id)
                BEGIN
                    DELETE FROM Topic
                    WHERE CrsID = @id
                END
            CREATE TABLE #questionIDS (
            questionID INT
            );
            insert into #questionIDS
            select id from Question where CrsID=@id;
```

```
            DECLARE @QID INT;

            DECLARE question_cursor CURSOR FOR
            SELECT questionID from #questionIDS

            OPEN question_cursor;

            FETCH NEXT FROM question_cursor INTO @QID;

            WHILE @@FETCH_STATUS = 0
                BEGIN
                Delete from Choice
                where  QID = @QID

                FETCH NEXT FROM question_cursor INTO @QID;
                END;

            CLOSE question_cursor;
            DEALLOCATE question_cursor;
            delete from Course where ID=@id;
END;
GO
GRANT EXECUTE ON  [dbo].[sp_DeleteCourse] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_DeleteCourse stored procedure deletes a
course and its related data. It first checks if the course exists by name. If found, it deletes
associated topics, question choices, and related questions before deleting the course itself. If
the course does not exist, it prints an error message.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_-
DeleteCourse', NULL, NULL
GO
```

Uses

[dbo].[Choice]
[dbo].[Course]
[dbo].[Question]
[dbo].[Topic]

[dbo].[sp_DeleteTopic]

MS_Description

The sp_DeleteTopic stored procedure deletes a topic from a specified course. It first checks if the topic and course exist. If they do, the topic is removed from the course. If either the topic or course is not found, an appropriate error message is printed.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@TopicName	nvarchar(100)	200
@CourseName	nvarchar(100)	200

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE PROCEDURE [dbo].[sp_DeleteTopic]
    @TopicName NVARCHAR(100),
    @CourseName NVARCHAR(100)
AS
BEGIN
    IF EXISTS (SELECT 1 FROM Topic WHERE name = @TopicName)and
        EXISTS (SELECT 1 FROM course WHERE name = @CourseName )
    DELETE FROM [dbo].[Topic]
    WHERE [CrsID] = (select id from Course where Name=@CourseName) and name =@TopicName;
    else if not EXISTS (SELECT 1 FROM Topic WHERE name = @TopicName)
     print 'No topic found with the specified name'
    else if not EXISTS (SELECT 1 FROM course WHERE name = @CourseName )
     print 'the course you want to remove the topic from does not exist '
END;
GO
GRANT EXECUTE ON  [dbo].[sp_DeleteTopic] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_DeleteTopic stored procedure deletes a
topic from a specified course. It first checks if the topic and course exist. If they do, the
```

```
topic is removed from the course. If either the topic or course is not found, an appropriate
error message is printed.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_DeleteTopic', NULL, NULL
GO
```

Uses

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

[dbo].[sp_get_all_courses]

MS_Description

The sp_get_all_courses stored procedure retrieves all records from the Course table, providing a list of all available courses.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_get_all_courses]
AS
BEGIN
    SELECT *
    FROM Course;
END;
GO
GRANT EXECUTE ON  [dbo].[sp_get_all_courses] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_get_all_courses stored procedure
retrieves all records from the Course table, providing a list of all available courses.',
'SCHEMA', N'dbo', 'PROCEDURE', N'sp_get_all_courses', NULL, NULL
GO
```

Uses

[dbo].[Course]

[dbo].[sp_get_all_instructors]

MS_Description

The sp_get_all_instructors stored procedure retrieves details of all instructors, including their personal information from the Person table and their hire date from the Instructor table, by performing an inner join between the two tables. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_get_all_instructors]
WITH ENCRYPTION
AS
BEGIN
    SELECT
        p.*,
        I.Hiredate
    FROM
        Person p
    INNER JOIN
        Instructor I
    ON
        p.ID = I.InsID;
END;
GO
GRANT EXECUTE ON  [dbo].[sp_get_all_instructors] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_get_all_instructors stored procedure
retrieves details of all instructors, including their personal information from the Person table
and their hire date from the Instructor table, by performing an inner join between the two
tables. The procedure is encrypted to secure its definition.', 'SCHEMA', N'dbo', 'PROCEDURE',
N'sp_get_all_instructors', NULL, NULL
GO
```

Uses

[dbo].[Instructor]
[dbo].[Person]
INSTRUCTOR

[dbo].[sp_get_all_persons]

MS_Description

The sp_get_all_persons stored procedure retrieves all the records from the Person table, returning the complete details of all individuals stored in the database. The procedure is encrypted to secure its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_get_all_persons]
WITH ENCRYPTION
AS
BEGIN
    SELECT
        *
    FROM
        Person
END;
exec sp_get_all_persons
GO
GRANT EXECUTE ON  [dbo].[sp_get_all_persons] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_get_all_persons stored procedure
retrieves all the records from the Person table, returning the complete details of all
individuals stored in the database. The procedure is encrypted to secure its definition.',
'SCHEMA', N'dbo', 'PROCEDURE', N'sp_get_all_persons', NULL, NULL
GO
```

Uses

[dbo].[Person]

[dbo].[sp_get_all_students]

MS_Description

The sp_get_all_students stored procedure retrieves all records from the Person table along with IntakeID and College details from the Student table, using an inner join on ID and StdID. The procedure is encrypted to protect its definition.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_get_all_students]
WITH ENCRYPTION
AS
BEGIN
    SELECT
        p.*,
        s.IntakeID,
        s.College
    FROM
        Person p
    INNER JOIN
        Student s
    ON
        p.ID = s.StdID;
END;
GO
GRANT EXECUTE ON  [dbo].[sp_get_all_students] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_get_all_students stored procedure
retrieves all records from the Person table along with IntakeID and College details from the
Student table, using an inner join on ID and StdID. The procedure is encrypted to protect its
definition.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_get_all_students', NULL, NULL
GO
```

Uses

[dbo].[Person]
[dbo].[Student]
STUDENT

[dbo].[sp_GET_exam_question_choice]

MS_Description

The sp_GET_exam_question_choice stored procedure retrieves the questions and choices for a specific exam identified by @examid. It checks if the user is an instructor or an admin before fetching the question and choice data. The procedure ensures the @examid is valid and returns choices for each question in the exam, or prints relevant messages for invalid inputs or access denial.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   procedure [dbo].[sp_GET_exam_question_choice]
@examid int
with encryption
as
begin
     declare @crsid int
     select @crsid=CrsID from exam where id=@examid
    IF @examid IS NULL OR LTRIM(RTRIM(@examid)) = ''
    BEGIN
        print 'Input cannot be empty or NULL.';
    END
    else IF NOT EXISTS (
        SELECT p.Email
        FROM Person p
        LEFT JOIN Ins_Crs ic ON p.ID = ic.InsID AND ic.CrsID = @crsid
        WHERE p.Email = SUSER_SNAME() AND (ic.InsID IS NOT NULL OR p.Role = 'admin')
```

```
    )
      BEGIN
            PRINT 'Access denied. You are not the instructor !.';
      END
   ELSE
    IF EXISTS (SELECT 1 FROM Exam WHERE ID = @examid)
   begin
    select  q.Body,
        MAX(CASE WHEN c.Choice = 1 THEN c.Body END) AS Choice1,
        MAX(CASE WHEN c.Choice = 2 THEN c.Body END) AS Choice2,
        MAX(CASE WHEN c.Choice = 3 THEN c.Body END) AS Choice3,
        MAX(CASE WHEN c.Choice = 4 THEN c.Body END) AS Choice4
        from Question q left join Choice c on q.ID=c.QID
        inner join Answer_Exam AE on AE.QID=q.ID and AE.ExamID=@examid group by q.Body
   end
end
GO
GRANT EXECUTE ON  [dbo].[sp_GET_exam_question_choice] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[sp_GET_exam_question_choice] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_GET_exam_question_choice stored procedure
retrieves the questions and choices for a specific exam identified by @examid. It checks if the
user is an instructor or an admin before fetching the question and choice data. The procedure
ensures the @examid is valid and returns choices for each question in the exam, or prints
relevant messages for invalid inputs or access denial.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_-
GET_exam_question_choice', NULL, NULL
GO
```

Uses

[dbo].[Answer_Exam]
[dbo].[Choice]
[dbo].[Exam]
[dbo].[Ins_Crs]
[dbo].[Person]
[dbo].[Question]

[dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER]

MS_Description

The sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER stored procedure retrieves questions and the student's answers for a specific exam identified by @examid and @stdID. It checks if the user has access rights (instructor or admin) and ensures that the exam and student exist. If valid, it returns the questions along with the student's answers from the Answer_Exam table.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@examid	int	4
@stdID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
create   procedure [dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER]
@examid int ,
@stdID int
with encryption
as
begin
     declare @crsid int
     select @crsid=CrsID from exam where id=@examid
     IF @examid IS NULL OR LTRIM(RTRIM(@examid)) = ''
       BEGIN
        print 'Input cannot be empty or NULL.';
       END
       else IF NOT EXISTS (
        SELECT p.Email
        FROM Person p
```

```
        LEFT JOIN Ins_Crs ic ON p.ID = ic.InsID AND ic.CrsID = 2
        WHERE p.Email = SUSER_SNAME() AND (ic.InsID IS NOT NULL OR p.Role = 'admin')
       )
      BEGIN
            PRINT 'Access denied. You are not the instructor !.';
      END
   ELSE
   begin
    IF EXISTS (SELECT 1 FROM Exam WHERE ID = @examid)
      begin
        select  q.Body,c.Body
        from Question q inner join Answer_Exam AE on q.ID=AE.QID
        inner join Choice c on c.Choice =AE.Answer and c.QID=q.ID where AE.ExamID=@examid
        and AE.StdID=@stdID
      end
    else
     print 'exam or student  maybe no exsists'
 end
end
GO
GRANT EXECUTE ON  [dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_GET_EXAM_QUESTIONS_WITH_STUDENT_ANSWER
stored procedure retrieves questions and the student''s answers for a specific exam identified by
@examid and @stdID. It checks if the user has access rights (instructor or admin) and ensures
that the exam and student exist. If valid, it returns the questions along with the student''s
answers from the Answer_Exam table.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_GET_EXAM_QUESTIONS_-
WITH_STUDENT_ANSWER', NULL, NULL
GO
```

Uses

[dbo].[Answer_Exam]
[dbo].[Choice]
[dbo].[Exam]
[dbo].[Ins_Crs]
[dbo].[Person]
[dbo].[Question]

[dbo].[sp_get_instructor_byID]

MS_Description

The sp_get_instructor_byID stored procedure retrieves the details of an instructor based on the provided @insID. It checks if the @insID is not null and exists in the Instructor table. If valid, it returns the instructor's personal details along with their hire date; otherwise, it prints an error message indicating the instructor does not exist.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_get_instructor_byID]
@insID int =null
WITH ENCRYPTION
AS
BEGIN
 IF @insID is null
           print 'instructor ID should not be null OR EMPTY'
 ELSE IF EXISTS (SELECT * FROM Instructor WHERE InsID = @insID)
 begin
    SELECT
        p.*,
        I.Hiredate
    FROM
        Person p
    INNER JOIN
        Instructor I
    ON
        p.ID = I.InsID and I.InsID=@insID;
```

```
 end
 else
   print 'the instructor does not exist in the instructor table '
END;
GO
GRANT EXECUTE ON  [dbo].[sp_get_instructor_byID] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_get_instructor_byID stored procedure
retrieves the details of an instructor based on the provided @insID. It checks if the @insID is
not null and exists in the Instructor table. If valid, it returns the instructor''s personal
details along with their hire date; otherwise, it prints an error message indicating the
instructor does not exist.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_get_instructor_byID', NULL, NULL
GO
```

Uses

[dbo].[Instructor]
[dbo].[Person]
INSTRUCTOR

[dbo].[sp_get_person_role]

MS_Description

The sp_get_person_role stored procedure retrieves the name and role of a person based on the provided @personID. It checks if the @personID is not null and exists in the Person table. If valid, it returns the person's name and role; otherwise, it prints an error message indicating the person does not exist.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_get_person_role]
@personID int
WITH ENCRYPTION
AS
BEGIN
    IF @personID is null
           print 'person ID should not be null OR EMPTY'
 ELSE IF EXISTS (SELECT * FROM Person WHERE ID = @personID)
 begin
    SELECT
        name,role
    FROM
        Person where ID=@personID;
 end
 else
   print 'the person does not exist in the person table '
END;
GO
```

```
GRANT EXECUTE ON  [dbo].[sp_get_person_role] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_get_person_role stored procedure
retrieves the name and role of a person based on the provided @personID. It checks if the
@personID is not null and exists in the Person table. If valid, it returns the person''s name and
role; otherwise, it prints an error message indicating the person does not exist.', 'SCHEMA',
N'dbo', 'PROCEDURE', N'sp_get_person_role', NULL, NULL
GO
```

Uses

[dbo].[Person]

[dbo].[sp_get_student_byID]

MS_Description

The sp_get_student_byID stored procedure retrieves the details of a student based on the provided @stdID. It checks if the @stdID is not null and exists in the Student table. If valid, it returns the student's personal details along with their IntakeID and College; otherwise, it prints an error message indicating the student does not exist.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_get_student_byID]
@stdID int =null
WITH ENCRYPTION
AS
BEGIN
 IF @stdID is null
           print 'student ID should not be null OR EMPTY'
 ELSE IF EXISTS (SELECT * FROM student WHERE StdID = @stdID)
 begin
    SELECT
        p.*,
        s.IntakeID,
        s.College
    FROM
        Person p
    INNER JOIN
```

```
        Student s
    ON
        p.ID = s.StdID and s.StdID=@stdID;
 end
 else
   print 'the student does not exist in the student table '
END;
GO
GRANT EXECUTE ON  [dbo].[sp_get_student_byID] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[sp_get_student_byID] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_get_student_byID stored procedure
retrieves the details of a student based on the provided @stdID. It checks if the @stdID is not
null and exists in the Student table. If valid, it returns the student''s personal details along
with their IntakeID and College; otherwise, it prints an error message indicating the student
does not exist.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_get_student_byID', NULL, NULL
GO
```

Uses

[dbo].[Person]
[dbo].[Student]
STUDENT

[dbo].[sp_GetAllCoursesWithTopics]

MS_Description

The sp_GetAllCoursesWithTopics stored procedure retrieves a list of all courses along with their associated topics. It returns the course name, the number of hours for each course, and a concatenated list of topics (or "no topic" if none are associated). The procedure uses a left join between the Course and Topic tables, grouping by course name and hours.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_GetAllCoursesWithTopics]
AS
BEGIN
    SELECT
        C.Name AS CourseName,
        C.Hours AS CourseHours,
        ISNULL( STRING_AGG(T.Name, ', '),'no topic') AS Topics
    FROM Course C
    LEFT JOIN Topic T ON C.ID = T.CrsID
    GROUP BY  C.Name, C.Hours;
END;
GO
GRANT EXECUTE ON  [dbo].[sp_GetAllCoursesWithTopics] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_GetAllCoursesWithTopics stored procedure
retrieves a list of all courses along with their associated topics. It returns the course name,
the number of hours for each course, and a concatenated list of topics (or "no topic" if none are
associated). The procedure uses a left join between the Course and Topic tables, grouping by
course name and hours.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_GetAllCoursesWithTopics', NULL, NULL
GO
```

Uses

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

[dbo].[sp_GetCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_GetCourse]
    @courseid int
AS
BEGIN
    IF @courseid IS NULL OR LTRIM(RTRIM(@courseid)) = ''
    BEGIN
        print 'course id cannot be empty or NULL.';
    END

  else IF not EXISTS (SELECT * FROM course c WHERE c.ID = @courseid)
        PRINT 'No course found with the specified name';
  ELSE IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN Ins_Crs ic ON p.ID = ic.InsID
WHERE ic.CrsID =@courseid))
        BEGIN
            PRINT 'Access denied. You are not the instructor for this course.';

        END

  else
    IF EXISTS (SELECT * FROM course c WHERE c.ID = @courseid)
        begin
            SELECT
            c.name AS courseName,c.Hours as courseHour,
```

```
            ISNULL(STRING_AGG(t.Name, ', '), 'No topics') AS Topics
        FROM
            course c
        left JOIN
            topic t ON c.ID = t.CrsID where c.ID=@courseid
            group by c.name,c.Hours
        end
    else
     print 'course maybe not exists!'
END;
GO
GRANT EXECUTE ON  [dbo].[sp_GetCourse] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[sp_GetCourse] TO [INSTRUCTOR]
GO
```

Uses

[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Person]
[dbo].[Topic]

[dbo].[sp_GetCourse_ITS_Topics]

MS_Description

The sp_GetCourse_ITS_Topics stored procedure retrieves a list of topic names associated with a specific course, identified by @CrsID. It selects the topic names from the Topic table where the CrsID matches the provided value. The procedure is executed by passing a specific CrsID (e.g., 1 in this case).

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	STUDENT
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE PROCEDURE [dbo].[sp_GetCourse_ITS_Topics]
    @CrsID INT
AS
BEGIN
    SELECT
        t.Name AS TopicName
    FROM Topic t
    WHERE t.CrsID = @CrsID;
END;
GO
GRANT EXECUTE ON  [dbo].[sp_GetCourse_ITS_Topics] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[sp_GetCourse_ITS_Topics] TO [INSTRUCTOR]
GO
GRANT EXECUTE ON  [dbo].[sp_GetCourse_ITS_Topics] TO [STUDENT]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_GetCourse_ITS_Topics stored procedure
```

```
retrieves a list of topic names associated with a specific course, identified by @CrsID. It
selects the topic names from the Topic table where the CrsID matches the provided value. The
procedure is executed by passing a specific CrsID (e.g., 1 in this case).', 'SCHEMA', N'dbo',
'PROCEDURE', N'sp_GetCourse_ITS_Topics', NULL, NULL
GO
```

Uses

[dbo].[Topic]

[dbo].[sp_std_courses_grade]

MS_Description

The sp_std_courses_grade stored procedure retrieves the grades of a student for completed exams. It takes @stdID as input and joins the Student_Exam, Exam, and Course tables to return the course name, exam name, and grade. It filters results by ensuring the exam's EndDate is before the current date, meaning only completed exams are included. The procedure is executed by passing a specific student ID (e.g., 3 in this case).

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	STUDENT

SQL Script

```
create   procedure [dbo].[sp_std_courses_grade]
@stdID int
with encryption
as
begin
select C.Name,E.Name,SE.grade from Student_Exam SE
inner join exam E on E.id=SE.examid inner join course C on C.ID =E.CrsID where SE.StdID=@stdID
and E.EndDate < GETDATE()
end
GO
GRANT EXECUTE ON  [dbo].[sp_std_courses_grade] TO [STUDENT]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_std_courses_grade stored procedure
retrieves the grades of a student for completed exams. It takes @stdID as input and joins the
Student_Exam, Exam, and Course tables to return the course name, exam name, and grade. It filters
results by ensuring the exam''s EndDate is before the current date, meaning only completed exams
are included. The procedure is executed by passing a specific student ID (e.g., 3 in this
case).', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_std_courses_grade', NULL, NULL
```

```
GO
```

Uses

[dbo].[Course]
[dbo].[Exam]
[dbo].[Student_Exam]

[dbo].[sp_std_courses_instructor]

MS_Description

The sp_std_courses_instructor stored procedure retrieves the list of courses taught by a specific instructor, identified by @instructID, along with the number of students enrolled in each course. It joins the Course, Ins_Crs, and Std_Crs tables to count the students per course. The procedure returns the course name and the corresponding student count, grouped by course name.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
create   procedure [dbo].[sp_std_courses_instructor]
@instructID int
with encryption
as
begin
select C.Name ,count(Sc.StdID) as 'stundent number' from course C inner join Ins_Crs IC
 on C.ID= IC.CrsID
left join Std_Crs SC on SC.StdID=c.ID and IC.InsID=@instructID group by c.Name
end
GO
GRANT EXECUTE ON  [dbo].[sp_std_courses_instructor] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[sp_std_courses_instructor] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_std_courses_instructor stored procedure
retrieves the list of courses taught by a specific instructor, identified by @instructID, along
with the number of students enrolled in each course. It joins the Course, Ins_Crs, and Std_Crs
```

```
tables to count the students per course. The procedure returns the course name and the
corresponding student count, grouped by course name.', 'SCHEMA', N'dbo', 'PROCEDURE',
N'sp_std_courses_instructor', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Std_Crs]

[dbo].[sp_std_exam_answers]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@examid	int	4
@stdID	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
create   procedure [dbo].[sp_std_exam_answers]
@examid int ,
@stdID int
with encryption
as
begin
select  q.Body,c.Body
      from Question q inner join Answer_Exam AE on q.ID=AE.QID
       inner join Choice c on c.Choice =AE.Answer and c.QID=q.ID where AE.ExamID=@examid
        and AE.StdID=@stdID
end
GO
GRANT EXECUTE ON  [dbo].[sp_std_exam_answers] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[sp_std_exam_answers] TO [INSTRUCTOR]
GO
```

Uses

[dbo].[Answer_Exam]
[dbo].[Choice]
[dbo].[Question]

[dbo].[sp_std_info_depart]

MS_Description

The sp_std_info_depart stored procedure retrieves the personal information of students belonging to a specific department, identified by @deptID. It joins the Person, Department, and Student tables to return the student's details along with their IntakeID and College, filtered by the department and the role being 'student'.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create   procedure [dbo].[sp_std_info_depart]
@deptID int
with encryption
as
begin
select  p.*,s.IntakeID,s.College from Person p inner join Department d on p.DeptID=d.ID and
p.Role='student'
and d.ID=@deptID
inner join Student s on s.StdID=p.ID
end
GO
GRANT EXECUTE ON  [dbo].[sp_std_info_depart] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_std_info_depart stored procedure
retrieves the personal information of students belonging to a specific department, identified by
@deptID. It joins the Person, Department, and Student tables to return the student''s details
along with their IntakeID and College, filtered by the department and the role being
''student''.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_std_info_depart', NULL, NULL
GO
```

Uses

[dbo].[Department]
[dbo].[Person]
[dbo].[Student]
STUDENT

[dbo].[sp_submit_exam_answer]

MS_Description

The sp_submit_exam_answer stored procedure allows a student to submit an answer for a specific exam question. It performs several validation checks, including verifying the student's enrollment in the course, checking the exam's start and end dates, and ensuring the student's answer hasn't been submitted already. If all conditions are met, it updates the student's answer in the Answer_Exam table and calculates the student's grade. The procedure ensures data integrity by using transactions and rolling back if any validation fails or errors occur.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@examID	int	4
@QID	int	4
@stdAnswer	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	STUDENT

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_submit_exam_answer]
@examID INT,
@QID INT,
@stdAnswer INT
WITH ENCRYPTION
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        DECLARE @stdemail NVARCHAR(MAX) = SUSER_SNAME();
        DECLARE @stdid INT;
        DECLARE @crsid INT;
        DECLARE @currentDate DATETIME = GETDATE();
```

```
        DECLARE @exam_Start_Date DATETIME;
        DECLARE @exam_end_Date DATETIME;
        SELECT @stdid = p.id FROM Person p WHERE p.email = @stdemail;
        SELECT @crsid = CrsID FROM std_crs WHERE StdID = @stdid;
        SELECT @exam_Start_Date = StartDate, @exam_end_Date = EndDate FROM Exam WHERE ID = @exam-
ID;
        IF NOT EXISTS (SELECT 1 FROM COURSE WHERE ID = @crsid)
        BEGIN
            PRINT 'Input Course ID cannot be found.';
            ROLLBACK TRANSACTION;
        END;
       else IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN std_Crs sc ON p.ID =
sc.stdid WHERE sc.CrsID =@crsid))
        BEGIN
            PRINT 'Access denied. You are not a student in this course.';
            ROLLBACK TRANSACTION;
        END;


        else IF @exam_end_Date < @currentDate
        BEGIN
            PRINT 'Your exam is finished. Sorry!';
            ROLLBACK TRANSACTION;
        END;
        else IF EXISTS (
            SELECT 1 FROM Answer_Exam
            WHERE ExamID = @examID AND QID = @QID AND StdID = @stdID
        ) AND @exam_Start_Date <= @currentDate
        BEGIN

            UPDATE Answer_Exam
            SET Answer = @stdAnswer
            WHERE ExamID = @examID AND QID = @QID AND StdID = @stdID;
            EXEC sp_calc_student_grade @stdid, @examID;

            PRINT 'Your answer has been submitted successfully.';
            COMMIT TRANSACTION;
        END
        ELSE
        BEGIN
            PRINT 'A problem occurred. Please re-enter your data or exam is not started!';
            ROLLBACK TRANSACTION;
        END;
    END TRY
    BEGIN CATCH
        PRINT 'An error occurred while submitting your exam answer.';
        ROLLBACK TRANSACTION;
    END CATCH;
END;

GO
GRANT EXECUTE ON  [dbo].[sp_submit_exam_answer] TO [STUDENT]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_submit_exam_answer stored procedure
```

```
allows a student to submit an answer for a specific exam question. It performs several validation
checks, including verifying the student''s enrollment in the course, checking the exam''s start
and end dates, and ensuring the student''s answer hasn''t been submitted already. If all
conditions are met, it updates the student''s answer in the Answer_Exam table and calculates the
student''s grade. The procedure ensures data integrity by using transactions and rolling back if
any validation fails or errors occur.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_submit_exam_answer',
NULL, NULL
GO
```

Uses

[dbo].[Answer_Exam]
[dbo].[Course]
[dbo].[Exam]
[dbo].[Person]
[dbo].[Std_Crs]
[dbo].[sp_calc_student_grade]

[dbo].[sp_UpdateCourseHour]

MS_Description

The sp_UpdateCourseHour stored procedure updates the number of hours for a course specified by its Name. It first checks if the course exists in the Course table. If no course is found, it prints a message. If the specified @Hour is not between 3 and 100, an error is raised. Otherwise, it updates the course's hours with the provided value.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Name	nvarchar(100)	200
@Hour	int	4

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_UpdateCourseHour]
    @Name NVARCHAR(100),
    @Hour int
AS
BEGIN
    declare @id int;
    select @id=id from Course where Name=@Name;
    IF @@ROWCOUNT = 0
        PRINT 'No course found with the specified name';
    else if (@Hour not between 3 and 100 )
      RAISERROR ('Course hours must be between 3 and 100.', 16, 1);
    else
       update course set Hours=@Hour where ID=@id
END;
GO
GRANT EXECUTE ON  [dbo].[sp_UpdateCourseHour] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_UpdateCourseHour stored procedure updates
```

```
the number of hours for a course specified by its Name. It first checks if the course exists in
the Course table. If no course is found, it prints a message. If the specified @Hour is not
between 3 and 100, an error is raised. Otherwise, it updates the course''s hours with the
provided value.', 'SCHEMA', N'dbo', 'PROCEDURE', N'sp_UpdateCourseHour', NULL, NULL
GO
```

Uses

[dbo].[Course]

[dbo].[sp_UpdateCourseName]

MS_Description

The sp_UpdateCourseName stored procedure updates the name of a course specified by @oldName to a new name provided as @newName. It first checks if the course exists in the Course table by searching for the @oldName. If the course is not found, it prints a message. Otherwise, it updates the course name with the new value.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@oldName	nvarchar(100)	200
@newName	nvarchar(100)	200

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROCEDURE [dbo].[sp_UpdateCourseName]
    @oldName NVARCHAR(100),
    @newName NVARCHAR(100)
AS
BEGIN
    declare @id int;
    select @id=id from Course where Name=@oldName;
    IF @@ROWCOUNT = 0
        PRINT 'No course found with the specified name';
    else
       update course set Name=@newName where ID=@id
END;
GO
GRANT EXECUTE ON  [dbo].[sp_UpdateCourseName] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_UpdateCourseName stored procedure updates
the name of a course specified by @oldName to a new name provided as @newName. It first checks if
the course exists in the Course table by searching for the @oldName. If the course is not found,
it prints a message. Otherwise, it updates the course name with the new value.', 'SCHEMA',
```

```
N'dbo', 'PROCEDURE', N'sp_UpdateCourseName', NULL, NULL
GO
```

Uses

[dbo].[Course]

[dbo].[sp_UpdateTopicName]

MS_Description

The sp_UpdateTopicName stored procedure updates the name of a topic from @oldTopicName to @newTopicName. It checks if the topic with the given @oldTopicName exists in the Topic table. If the topic is found, it updates the topic's name. Otherwise, it prints a message indicating that no topic with the specified name was found.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@oldTopicName	nvarchar(100)	200
@newTopicName	nvarchar(100)	200

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE PROCEDURE [dbo].[sp_UpdateTopicName]
    @oldTopicName NVARCHAR(100),
    @newTopicName NVARCHAR(100)
AS
BEGIN
    IF EXISTS (SELECT 1 FROM Topic WHERE name = @oldTopicName)
    begin
        UPDATE [dbo].[Topic]
        SET [Name] = @newTopicName
        WHERE [Name] = @oldTopicName;
    end
    else
     print 'No topic found with the specified name'
END;
GO
GRANT EXECUTE ON  [dbo].[sp_UpdateTopicName] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The sp_UpdateTopicName stored procedure updates
```

```
the name of a topic from @oldTopicName to @newTopicName. It checks if the topic with the given
@oldTopicName exists in the Topic table. If the topic is found, it updates the topic''s name.
Otherwise, it prints a message indicating that no topic with the specified name was found.',
'SCHEMA', N'dbo', 'PROCEDURE', N'sp_UpdateTopicName', NULL, NULL
GO
```

Uses

[dbo].[Topic]

[dbo].[updateDepartmentName]

MS_Description

The updateDepartmentName stored procedure updates the name and description of a department based on the provided @DID (Department ID). It first checks if the @DID or @newName is null or empty. If any of these are invalid, it prints an error message. Then, it verifies if the department exists using the @DID. If the department exists, it updates the department's name and description; otherwise, it prints a message indicating the department doesn't exist.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@DID	int	4
@newName	nvarchar(100)	200
@Describe	nvarchar(255)	510

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE PROCEDURE [dbo].[updateDepartmentName]
    @DID INT, @newName NVARCHAR(100) , @Describe NVARCHAR(255)
With encryption
AS
BEGIN
    IF @DID IS NULL OR LTRIM(RTRIM(@DID)) = ''
    and @newName is null OR LTRIM(RTRIM(@DID)) = ''
    BEGIN
        print 'input cannot be empty or NULL.';
    END

  else IF EXISTS (SELECT 1 FROM Department WHERE ID = @DID)
     begin
       UPDATE Department SET Name = @newName  , Describe=@Describe
```

```
        WHERE ID = @DID
        print 'Update intake name successfully.';
      end

    ELSE
    begin
      print 'the intake name is not exists'
    end
end;
GO
GRANT EXECUTE ON  [dbo].[updateDepartmentName] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The updateDepartmentName stored procedure
updates the name and description of a department based on the provided @DID (Department ID). It
first checks if the @DID or @newName is null or empty. If any of these are invalid, it prints an
error message. Then, it verifies if the department exists using the @DID. If the department
exists, it updates the department''s name and description; otherwise, it prints a message
indicating the department doesn''t exist.', 'SCHEMA', N'dbo', 'PROCEDURE', N'updateDepartment-
Name', NULL, NULL
GO
```

Uses

[dbo].[Department]

[dbo].[updateExamStartDate]

MS_Description

The updateExamStartDate stored procedure updates the start date of an exam identified by @EX_id to a new date provided by @newdate. It checks if the @EX_id is valid and whether the new start date is in the future. The procedure also ensures that the user has instructor-level access for the course associated with the exam. If the exam exists, the start date is updated; otherwise, it prints an error message. It includes error handling to manage potential issues during execution.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

Name	Data Type	Max Length (Bytes)
@EX_id	int	4
@newdate	datetime	8

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR

SQL Script

```
CREATE   PROCEDURE [dbo].[updateExamStartDate]
    @EX_id INT, @newdate DATETIME
With encryption
AS
BEGIN
   begin try
   declare @crsid int
   select @crsid=CrsID from exam where id =@EX_id
    IF @EX_id is null
        print 'Exam id cannot be empty or NULL.'
    else IF @newdate <= GETDATE()
            PRINT 'StartDate cannot be in the past or today or empty.';
    ELSE IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN Ins_Crs ic ON p.ID =
ic.InsID WHERE ic.CrsID =@crsid ))
        BEGIN
            PRINT 'Access denied. You are not the instructor !.';
```

```
        END
   ELSE
    begin
      IF EXISTS (SELECT 1 FROM EXAM WHERE ID = @EX_id)
     begin
       UPDATE Exam SET StartDate = @newdate
       WHERE ID = @EX_id
       print 'Update exam start date successfully.';
     end
     else
        PRINT 'Exam with the given ID does not exist.';
    end
  end try
  begin catch
      print 'there are error occured'
  end catch

end;
GO
GRANT EXECUTE ON  [dbo].[updateExamStartDate] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The updateExamStartDate stored procedure updates
the start date of an exam identified by @EX_id to a new date provided by @newdate. It checks if
the @EX_id is valid and whether the new start date is in the future. The procedure also ensures
that the user has instructor-level access for the course associated with the exam. If the exam
exists, the start date is updated; otherwise, it prints an error message. It includes error
handling to manage potential issues during execution.', 'SCHEMA', N'dbo', 'PROCEDURE', N'update-
ExamStartDate', NULL, NULL
GO
```

Uses

[dbo].[Exam]
[dbo].[Ins_Crs]
[dbo].[Person]

[dbo].[updateIntakeName]

MS_Description

The updateIntakeName stored procedure updates the name of an intake from @oldName to @newName. It first checks if the @oldName is not null or empty. Then, it verifies if the intake with the given @oldName exists in the Intake table. If the intake exists, it updates the name to @newName; otherwise, it prints an error message indicating the intake doesn't exist.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@oldName	nvarchar(100)	200
@newName	nvarchar(100)	200

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create   PROCEDURE [dbo].[updateIntakeName]
    @oldName NVARCHAR(100), @newName NVARCHAR(100)
AS
BEGIN
    IF @oldName IS NULL OR LTRIM(RTRIM(@oldName)) = ''
    BEGIN
        print 'Name cannot be empty or NULL.';
    END

  else IF EXISTS (SELECT 1 FROM Intake WHERE Name = @oldName)
     begin
       UPDATE Intake SET Name = @newName
       WHERE Name = @oldName
       print 'Update intake name successfully.';
     end

    ELSE
    begin
```

```
      print 'the intake name is not exists'
    end
end;
GO
GRANT EXECUTE ON  [dbo].[updateIntakeName] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The updateIntakeName stored procedure updates
the name of an intake from @oldName to @newName. It first checks if the @oldName is not null or
empty. Then, it verifies if the intake with the given @oldName exists in the Intake table. If the
intake exists, it updates the name to @newName; otherwise, it prints an error message indicating
the intake doesn''t exist.', 'SCHEMA', N'dbo', 'PROCEDURE', N'updateIntakeName', NULL, NULL
GO
```

Uses

[dbo].[Intake]

[dbo].[updateIntakeNameAndDate]

MS_Description

The updateIntakeNameAndDate stored procedure updates both the name and the start date of an intake specified by @oldName. It first checks if the @oldName is not null or empty. Then, it verifies if an intake with the given name exists. If the intake is found, it starts a transaction to update the intake's name and start date to the provided values (@NewName and @New_date). If any error occurs during the process, it rolls back the transaction and prints an error message. If the intake name doesn't exist, it prints a message indicating so.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@oldName	nvarchar(100)	200
@NewName	nvarchar(100)	200
@New_date	date	3

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create PROCEDURE [dbo].[updateIntakeNameAndDate]
    @oldName NVARCHAR(100),
    @NewName NVARCHAR(100),
    @New_date DATE
AS
BEGIN
    -- Check if the old name is NULL or empty
    IF @oldName IS NULL OR LTRIM(RTRIM(@oldName)) = ''
    BEGIN
        PRINT 'Name cannot be empty or NULL.';
        RETURN; -- Exit procedure if old name is invalid
    END

    -- Check if the intake with the old name exists
    IF EXISTS (SELECT 1 FROM Intake WHERE Name = @oldName)
```

```
    BEGIN
        BEGIN TRANSACTION;
            BEGIN TRY
                -- Update the intake record with new name and date
                UPDATE Intake
                SET StartDate = @New_date, Name = @NewName
                WHERE Name = @oldName;

                COMMIT TRANSACTION;
                PRINT 'Update intake name ' + @oldName + ' to ' + @NewName + ' and start date to
' + CONVERT(NVARCHAR(20), @New_date, 120) + ' successfully.';
            END TRY
            BEGIN CATCH
                -- Rollback if an error occurs
                ROLLBACK TRANSACTION;
                PRINT 'An error occurred: please check again on your data may intake name do not
follow rules ';
            END CATCH
    END
    ELSE
    BEGIN
        PRINT 'The intake name does not exist.';
    END
END;
GO
GRANT EXECUTE ON  [dbo].[updateIntakeNameAndDate] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The updateIntakeNameAndDate stored procedure
updates both the name and the start date of an intake specified by @oldName. It first checks if
the @oldName is not null or empty. Then, it verifies if an intake with the given name exists. If
the intake is found, it starts a transaction to update the intake''s name and start date to the
provided values (@NewName and @New_date). If any error occurs during the process, it rolls back
the transaction and prints an error message. If the intake name doesn''t exist, it prints a
message indicating so.', 'SCHEMA', N'dbo', 'PROCEDURE', N'updateIntakeNameAndDate', NULL, NULL
GO
```

Uses

[dbo].[Intake]

[dbo].[updateIntakeStartDate]

MS_Description

The updateIntakeStartDate stored procedure updates the start date of an intake identified by @intake_Name to the provided @St_date. It first checks if the @intake_Name is not null or empty. Then, it verifies if the intake with the given name exists. If the intake exists, it updates the start date. Otherwise, it prints a message indicating the intake does not exist.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@intake_Name	nvarchar(100)	200
@St_date	date	3

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create   PROCEDURE [dbo].[updateIntakeStartDate]
    @intake_Name NVARCHAR(100), @St_date date
AS
BEGIN
    IF @intake_Name IS NULL OR LTRIM(RTRIM(@intake_Name)) = ''
    BEGIN
        print 'Name cannot be empty or NULL.';
    END

  else IF EXISTS (SELECT 1 FROM Intake WHERE Name = @intake_Name)
     begin
       UPDATE Intake SET StartDate = @St_date
       WHERE Name = @intake_Name
       print 'Update Start date for intake ' + @intake_Name +  ' successfully.';
     end

    ELSE
    begin
```

```
      print 'the intake name is not exists.'
    end
end;
GO
GRANT EXECUTE ON  [dbo].[updateIntakeStartDate] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The updateIntakeStartDate stored procedure
updates the start date of an intake identified by @intake_Name to the provided @St_date. It first
checks if the @intake_Name is not null or empty. Then, it verifies if the intake with the given
name exists. If the intake exists, it updates the start date. Otherwise, it prints a message
indicating the intake does not exist.', 'SCHEMA', N'dbo', 'PROCEDURE', N'updateIntakeStartDate',
NULL, NULL
GO
```

Uses

[dbo].[Intake]

[dbo].[view_allQuestions_bycourse]

MS_Description

The view_allQuestions_bycourse stored procedure retrieves all questions and their associated choices for a given course specified by @CrsName. It first checks if the course exists and if the user has instructor-level access for that course. If the course is not found or the user is not an instructor for the course, an appropriate message is displayed. If the course exists and the user is authorized, the procedure returns the questions along with their possible choices (Choice1, Choice2, etc.) from the Question and Choice tables.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@CrsName	nvarchar(max)	max

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	INSTRUCTOR

SQL Script

```
CREATE   PROCEDURE [dbo].[view_allQuestions_bycourse]
@CrsName NVARCHAR(MAX)
As
begin


    declare @CrsID int;
    select @CrsID=id from Course where Name=@CrsName;
    IF @@ROWCOUNT = 0
       PRINT 'No course found with the specified name';
    else IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN Ins_Crs ic ON p.ID =
ic.InsID WHERE ic.CrsID =@CrsID))
        BEGIN
            PRINT 'Access denied. You are not the instructor for this course.';
        END
    ELSE
    begin
    SELECT
```

```
            q.Body AS QuestionBody,
            MAX(CASE WHEN c.Choice = 1 THEN c.body END) AS Choice1,
            MAX(CASE WHEN c.Choice = 2 THEN c.body END) AS Choice2,
            MAX(CASE WHEN c.Choice = 3 THEN c.body END) AS Choice3,
            MAX(CASE WHEN c.Choice = 4 THEN c.body END) AS Choice4
        FROM Question q
        inner JOIN
            Choice c ON q.ID = c.QID and q.CrsID = @CrsID
        GROUP BY
            q.ID, q.Body;
end
end;
GO
GRANT EXECUTE ON  [dbo].[view_allQuestions_bycourse] TO [INSTRUCTOR]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The view_allQuestions_bycourse stored procedure
retrieves all questions and their associated choices for a given course specified by @CrsName. It
first checks if the course exists and if the user has instructor-level access for that course. If
the course is not found or the user is not an instructor for the course, an appropriate message
is displayed. If the course exists and the user is authorized, the procedure returns the
questions along with their possible choices (Choice1, Choice2, etc.) from the Question and Choice
tables.', 'SCHEMA', N'dbo', 'PROCEDURE', N'view_allQuestions_bycourse', NULL, NULL
GO
```

Uses

[dbo].[Choice]
[dbo].[Course]
[dbo].[Ins_Crs]
[dbo].[Person]
[dbo].[Question]

[dbo].[view_allQuestionsByAdmin]

MS_Description

The view_allQuestionsByAdmin stored procedure retrieves all questions along with their associated choices (Choice1, Choice2, etc.) for all courses in the system, without any course or instructor restrictions. The procedure selects the question body and the possible choices from the Question and Choice tables and groups the results by question ID and body. The procedure is encrypted for security purposes.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create proc [dbo].[view_allQuestionsByAdmin]
with encryption
as
begin
       SELECT
            q.Body AS QuestionBody,
            MAX(CASE WHEN c.Choice = 1 THEN c.body END) AS Choice1,
            MAX(CASE WHEN c.Choice = 2 THEN c.body END) AS Choice2,
            MAX(CASE WHEN c.Choice = 3 THEN c.body END) AS Choice3,
            MAX(CASE WHEN c.Choice = 4 THEN c.body END) AS Choice4
        FROM Question q
        inner JOIN
            Choice c ON q.ID = c.QID
        GROUP BY
            q.ID, q.Body;
end
GO
GRANT EXECUTE ON  [dbo].[view_allQuestionsByAdmin] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The view_allQuestionsByAdmin stored procedure
retrieves all questions along with their associated choices (Choice1, Choice2, etc.) for all
courses in the system, without any course or instructor restrictions. The procedure selects the
question body and the possible choices from the Question and Choice tables and groups the results
by question ID and body. The procedure is encrypted for security purposes.', 'SCHEMA', N'dbo',
'PROCEDURE', N'view_allQuestionsByAdmin', NULL, NULL
```

```
GO
```

Uses

[dbo].[Choice]
[dbo].[Question]

[dbo].[ViewADepartmentsInfo]

MS_Description

The ViewADepartmentsInfo stored procedure retrieves all records from the Department table, displaying all available department details in the system. This procedure is encrypted for security purposes.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
Create Proc [dbo].[ViewADepartmentsInfo]
with encryption
as
  begin
      select * from  Department
  end
GO
GRANT EXECUTE ON  [dbo].[ViewADepartmentsInfo] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ViewADepartmentsInfo stored procedure
retrieves all records from the Department table, displaying all available department details in
the system. This procedure is encrypted for security purposes.', 'SCHEMA', N'dbo', 'PROCEDURE',
N'ViewADepartmentsInfo', NULL, NULL
GO
```

Uses

[dbo].[Department]

[dbo].[ViewAIntakesInfo]

MS_Description

The ViewAIntakesInfo stored procedure retrieves all records from the Intake table, displaying the details of all intakes in the system. This procedure is encrypted for security purposes.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
Create Proc [dbo].[ViewAIntakesInfo]
with encryption
as
  begin
      select * from  intake
  end
GO
GRANT EXECUTE ON  [dbo].[ViewAIntakesInfo] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ViewAIntakesInfo stored procedure retrieves
all records from the Intake table, displaying the details of all intakes in the system. This
procedure is encrypted for security purposes.', 'SCHEMA', N'dbo', 'PROCEDURE', N'ViewAIntakes-
Info', NULL, NULL
GO
```

Uses

[dbo].[Intake]

[dbo].[ViewAllInstructorCourse]

MS_Description

The ViewAllInstructorCourse stored procedure retrieves details about all courses assigned to instructors. It returns the instructor's ID, name, course name, and course ID by joining the Ins_Crs, Person, and Course tables. This procedure is encrypted for security purposes.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE PROC [dbo].[ViewAllInstructorCourse]
WITH ENCRYPTION
AS
  BEGIN
         select ic.InsID as [instructorID],p.Name as[Instructor_Name]  , c.Name as [Cpurse_Name]
,ic.CrsID
       from Ins_Crs ic join person p
       on p.ID=ic.InsID
       join Course c on  c.ID=ic.CrsID
  END
GO
GRANT EXECUTE ON  [dbo].[ViewAllInstructorCourse] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ViewAllInstructorCourse stored procedure
retrieves details about all courses assigned to instructors. It returns the instructor''s ID,
name, course name, and course ID by joining the Ins_Crs, Person, and Course tables. This
procedure is encrypted for security purposes.', 'SCHEMA', N'dbo', 'PROCEDURE', N'ViewAll-
InstructorCourse', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Ins_Crs]

[dbo].[Person]

[dbo].[ViewCoursesByStudent]

MS_Description

The ViewCoursesByStudent stored procedure retrieves courses associated with a specific student. If the student ID is provided, it checks if the student exists in the Std_Crs table. If the student exists, it returns their ID, name, the course ID, and course name. Otherwise, it prints an error message if the student ID is not found or if the ID is null.

This procedure ensures that only valid student IDs are processed, and course details are fetched accordingly.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	STUDENT
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE   PROC [dbo].[ViewCoursesByStudent]
    @St_id int=null
WITH ENCRYPTION
AS
 begin
 if @St_id is null
        print 'student ID should not be null OR EMPTY'
 ELSE IF EXISTS (SELECT * FROM Std_Crs WHERE StdID = @St_id)
       BEGIN
          SELECT s.StdID as [student_id] ,p.Name as [student_name]  ,s.CrsID as
[course_id],c.Name as [course_name]
          FROM Std_Crs s join person p
          on s.StdID=p.id
          join Course c on CrsID=c.ID
         where StdID=@St_id
        END
```

```
 else
   begin
     print 'student id: '+CAST(@St_id AS NVARCHAR(10))+ ' not fount'
   end
end
GO
GRANT EXECUTE ON  [dbo].[ViewCoursesByStudent] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[ViewCoursesByStudent] TO [STUDENT]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ViewCoursesByStudent stored procedure
retrieves courses associated with a specific student. If the student ID is provided, it checks if
the student exists in the Std_Crs table. If the student exists, it returns their ID, name, the
course ID, and course name. Otherwise, it prints an error message if the student ID is not found
or if the ID is null.
This procedure ensures that only valid student IDs are processed, and course details are fetched
accordingly.', 'SCHEMA', N'dbo', 'PROCEDURE', N'ViewCoursesByStudent', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Person]
[dbo].[Std_Crs]

[dbo].[ViewExamsInfo]

MS_Description

The ViewExamsInfo stored procedure retrieves all records from the Exam table. It doesn't require any input parameters and will simply return all the columns of the Exam table when executed. This procedure is helpful for administrators or instructors who need to view the complete list of exams.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	STUDENT
Grant	EXECUTE	ADMIN

SQL Script

```
CREATE Proc [dbo].[ViewExamsInfo]
with encryption
as
  begin
      select * from  Exam
  end
GO
GRANT EXECUTE ON  [dbo].[ViewExamsInfo] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[ViewExamsInfo] TO [STUDENT]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ViewExamsInfo stored procedure retrieves all
records from the Exam table. It doesn''t require any input parameters and will simply return all
the columns of the Exam table when executed. This procedure is helpful for administrators or
instructors who need to view the complete list of exams.', 'SCHEMA', N'dbo', 'PROCEDURE', N'View-
ExamsInfo', NULL, NULL
GO
```

Uses

[dbo].[Exam]

[dbo].[ViewSpecificDepartmentInfo]

MS_Description

The ViewSpecificDepartmentInfo procedure retrieves details of a department by its ID (@DID). It checks if the department exists, and if so, returns its information. If the ID is invalid or not found, an appropriate message is displayed. This is useful for administrators or users looking for specific department data.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create PROCEDURE [dbo].[ViewSpecificDepartmentInfo]
    @DID INT
AS
BEGIN
    IF @DID IS NULL OR LTRIM(RTRIM(@DID)) = ''
    BEGIN
        print 'Department id cannot be empty or NULL.';
    END

  else IF EXISTS (SELECT 1 FROM Department WHERE ID = @DID)
     begin
       select * from Department
       where ID=@DID

     end

    ELSE
    begin
      print 'the Department is not exists.'
```

```
    end
end;
GO
GRANT EXECUTE ON  [dbo].[ViewSpecificDepartmentInfo] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ViewSpecificDepartmentInfo procedure
retrieves details of a department by its ID (@DID). It checks if the department exists, and if
so, returns its information. If the ID is invalid or not found, an appropriate message is
displayed. This is useful for administrators or users looking for specific department data.',
'SCHEMA', N'dbo', 'PROCEDURE', N'ViewSpecificDepartmentInfo', NULL, NULL
GO
```

Uses

[dbo].[Department]

[dbo].[ViewSpecificExamInfo]

MS_Description

The ViewSpecificExamInfo stored procedure retrieves details of a specific exam based on the provided @EX_ID (exam ID). It checks if the exam exists and if the user is authorized (i.e., the user is the instructor for the related course). If the exam ID is not valid or the user lacks permission, an appropriate message is displayed. Otherwise, the exam information is returned. This procedure is useful for instructors or administrators who need to view the details of a particular exam.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

Parameters

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

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	STUDENT
Grant	EXECUTE	INSTRUCTOR
Grant	EXECUTE	ADMIN

SQL Script

```
create   PROCEDURE [dbo].[ViewSpecificExamInfo]
    @EX_ID INT
WITH ENCRYPTION
AS
BEGIN
    declare @crsid int
    select @crsid=CrsID from exam where id =@EX_ID
    IF @EX_ID IS NULL OR LTRIM(RTRIM(@EX_ID)) = ''
    BEGIN
        print 'Department id cannot be empty or NULL.';
    END
   ELSE IF (SUSER_SNAME() not in (SELECT  p.Email FROM Person p JOIN Ins_Crs ic ON p.ID = ic.Ins-
ID WHERE ic.CrsID =@crsid ))
        BEGIN
            PRINT 'Access denied. You are not the instructor for this exam !.';
```

```
        END
    ELSE
    begin
     IF EXISTS (SELECT 1 FROM Exam WHERE ID = @EX_ID)
     begin
       select * from exam
       where ID=@EX_ID
     end

    ELSE
    begin
      print 'the EXAM is not exists.'
    end
 end
end;
GO
GRANT EXECUTE ON  [dbo].[ViewSpecificExamInfo] TO [ADMIN]
GO
GRANT EXECUTE ON  [dbo].[ViewSpecificExamInfo] TO [INSTRUCTOR]
GO
GRANT EXECUTE ON  [dbo].[ViewSpecificExamInfo] TO [STUDENT]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ViewSpecificExamInfo stored procedure
retrieves details of a specific exam based on the provided @EX_ID (exam ID). It checks if the
exam exists and if the user is authorized (i.e., the user is the instructor for the related
course). If the exam ID is not valid or the user lacks permission, an appropriate message is
displayed. Otherwise, the exam information is returned. This procedure is useful for instructors
or administrators who need to view the details of a particular exam.', 'SCHEMA', N'dbo',
'PROCEDURE', N'ViewSpecificExamInfo', NULL, NULL
GO
```

Uses

[dbo].[Exam]
[dbo].[Ins_Crs]
[dbo].[Person]

[dbo].[ViewSpecificIntakeInfo]

MS_Description

The ViewSpecificIntakeInfo stored procedure retrieves the details of a specific intake based on the provided @intake_Name. It checks if the intake name is valid and exists in the Intake table. If the intake name is not found or is empty, an appropriate message is displayed. Otherwise, the procedure returns all information for the matching intake. This procedure is useful for administrators or department staff who need to view details of a particular intake.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@intake_Name	nvarchar(100)	200

Permissions

Type	Action	Owning Principal
Grant	EXECUTE	ADMIN

SQL Script

```
create   PROCEDURE [dbo].[ViewSpecificIntakeInfo]
    @intake_Name NVARCHAR(100)
AS
BEGIN
    IF @intake_Name IS NULL OR LTRIM(RTRIM(@intake_Name)) = ''
    BEGIN
        print 'Name cannot be empty or NULL.';
    END

  else IF EXISTS (SELECT 1 FROM Intake WHERE Name = @intake_Name)
     begin
       select * from intake
       where Name=@intake_Name

     end

    ELSE
    begin
```

```
      print 'the intake name is not exists.'
    end
end;
GO
GRANT EXECUTE ON  [dbo].[ViewSpecificIntakeInfo] TO [ADMIN]
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ViewSpecificIntakeInfo stored procedure
retrieves the details of a specific intake based on the provided @intake_Name. It checks if the
intake name is valid and exists in the Intake table. If the intake name is not found or is empty,
an appropriate message is displayed. Otherwise, the procedure returns all information for the
matching intake. This procedure is useful for administrators or department staff who need to view
details of a particular intake.', 'SCHEMA', N'dbo', 'PROCEDURE', N'ViewSpecificIntakeInfo', NULL,
NULL
GO
```

Uses

[dbo].[Intake]

[dbo].[ViewStudentCourse]

MS_Description

The ViewStudentCourse stored procedure retrieves the list of students and the courses they are enrolled in. It provides information such as the student ID, student name, course ID, and course name. This procedure is useful for administrators or instructors who want to view the enrollment details of students in different courses. The query joins the Std_Crs, Person, and Course tables to return the relevant data.

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Encrypted	True

SQL Script

```
CREATE PROC [dbo].[ViewStudentCourse]
WITH ENCRYPTION
AS

   BEGIN
      SELECT s.StdID as [student_id] ,p.Name as [student_name]  ,s.CrsID as [course_id],c.Name as
[course_name]
      FROM Std_Crs s join person p
      on s.StdID=p.id
      join Course c on CrsID=c.ID
   END
GO
EXEC sp_addextendedproperty N'MS_Description', N'The ViewStudentCourse stored procedure retrieves
the list of students and the courses they are enrolled in. It provides information such as the
student ID, student name, course ID, and course name. This procedure is useful for administrators
or instructors who want to view the enrollment details of students in different courses. The
query joins the Std_Crs, Person, and Course tables to return the relevant data.', 'SCHEMA',
N'dbo', 'PROCEDURE', N'ViewStudentCourse', NULL, NULL
GO
```

Uses

[dbo].[Course]
[dbo].[Person]
[dbo].[Std_Crs]

[instructor].[GET_PERSON]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
CREATE PROCEDURE [instructor].[GET_PERSON]
WITH EXECUTE AS CALLER
AS
BEGIN
    -- Select all rows from the dbo.Person table
    SELECT * FROM dbo.Person;
END;
GO
```

Uses

[dbo].[Person]
instructor

Users

Objects

Name
admin@example.com
dbo
duplicate@example.com
duplicate8@example.com
EBTIHAL.DOE@example.com
EBTIHAL@example.com
guest
invalid3.dept@example.com
jane.WILLIAN@example.com
MUATEF30@example.com
MUATEF300@example.com
MUATEF38@example.com
tefa@gmail.com
yasmena9000@example.com

admin@example.com

Properties

Property	Value
Type	SqlUser
Login Name	admin@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [admin@example.com] FOR LOGIN [admin@example.com]
GO
```

Used By

ADMIN

dbo

Properties

Property	Value
Type	SqlUser
Login Name	sa
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
GO
```

duplicate@example.com

Properties

Property	Value
Type	SqlUser
Login Name	duplicate@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [duplicate@example.com] FOR LOGIN [duplicate@example.com]
GO
```

Used By

STUDENT

duplicate8@example.com

Properties

Property	Value
Type	SqlUser
Login Name	duplicate8@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [duplicate8@example.com] FOR LOGIN [duplicate8@example.com]
GO
```

Used By

STUDENT

EBTIHAL.DOE@example.com

Properties

Property	Value
Type	SqlUser
Login Name	EBTIHAL.DOE@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [EBTIHAL.DOE@example.com] FOR LOGIN [EBTIHAL.DOE@example.com]
GO
```

Used By

STUDENT

EBTIHAL@example.com

Properties

Property	Value
Type	SqlUser
Login Name	EBTIHAL@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [EBTIHAL@example.com] FOR LOGIN [EBTIHAL@example.com]
GO
```

Used By

ADMIN

guest

Properties

Property	Value
Type	SqlUser
Default Schema	guest

SQL Script

```
GO
```

invalid3.dept@example.com

Properties

Property	Value
Type	SqlUser
Login Name	invalid3.dept@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [invalid3.dept@example.com] FOR LOGIN [invalid3.dept@example.com]
GO
```

Used By

STUDENT

jane.WILLIAN@example.com

Properties

Property	Value
Type	SqlUser
Login Name	jane.WILLIAN@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [jane.WILLIAN@example.com] FOR LOGIN [jane.WILLIAN@example.com]
GO
```

Used By

INSTRUCTOR

MUATEF30@example.com

Properties

Property	Value
Type	SqlUser
Login Name	MUATEF30@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [MUATEF30@example.com] FOR LOGIN [MUATEF30@example.com]
GO
```

Used By

STUDENT

MUATEF300@example.com

Properties

Property	Value
Type	SqlUser
Login Name	MUATEF300@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [MUATEF300@example.com] FOR LOGIN [MUATEF300@example.com]
GO
```

Used By

ADMIN

MUATEF38@example.com

Properties

Property	Value
Type	SqlUser
Login Name	MUATEF38@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [MUATEF38@example.com] FOR LOGIN [MUATEF38@example.com]
GO
```

Used By

INSTRUCTOR

tefa@gmail.com

Properties

Property	Value
Type	SqlUser
Login Name	tefa@gmail.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [tefa@gmail.com] FOR LOGIN [tefa@gmail.com]
GO
```

Used By

ADMIN

yasmena9000@example.com

Properties

Property	Value
Type	SqlUser
Login Name	yasmena9000@example.com
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [yasmena9000@example.com] FOR LOGIN [yasmena9000@example.com]
GO
```

Used By

ADMIN

Database Roles

Objects

Name
ADMIN
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
INSTRUCTOR
public
STUDENT

ADMIN

Properties

Property	Value
Owner	dbo

Members

- admin@example.com
- EBTIHAL@example.com
- MUATEF300@example.com
- tefa@gmail.com
- yasmena9000@example.com

SQL Script

```
CREATE ROLE [ADMIN]
AUTHORIZATION [dbo]
GO
ALTER ROLE [ADMIN] ADD MEMBER [admin@example.com]
GO
ALTER ROLE [ADMIN] ADD MEMBER [EBTIHAL@example.com]
GO
ALTER ROLE [ADMIN] ADD MEMBER [MUATEF300@example.com]
GO
ALTER ROLE [ADMIN] ADD MEMBER [tefa@gmail.com]
GO
ALTER ROLE [ADMIN] ADD MEMBER [yasmena9000@example.com]
GO
```

Uses

admin@example.com
EBTIHAL@example.com
MUATEF300@example.com
tefa@gmail.com
yasmena9000@example.com

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

INSTRUCTOR

Properties

Property	Value
Owner	dbo

Members

- jane.WILLIAN@example.com
- MUATEF38@example.com

SQL Script

```
CREATE ROLE [INSTRUCTOR]
AUTHORIZATION [dbo]
GO
ALTER ROLE [INSTRUCTOR] ADD MEMBER [jane.WILLIAN@example.com]
GO
ALTER ROLE [INSTRUCTOR] ADD MEMBER [MUATEF38@example.com]
GO
```

Uses

jane.WILLIAN@example.com
MUATEF38@example.com

Used By

[dbo].[InsertInstructorCourse]
[dbo].[ModifyCourseInstructor]
[dbo].[ModifyInstructorCourse]
[dbo].[sp_add_person]
[dbo].[sp_get_all_instructors]
[dbo].[sp_get_instructor_byID]

public

Properties

Property	Value
Owner	dbo

STUDENT

Properties

Property	Value
Owner	dbo

Members

- duplicate@example.com
- duplicate8@example.com
- EBTIHAL.DOE@example.com
- invalid3.dept@example.com
- MUATEF30@example.com

SQL Script

```
CREATE ROLE [STUDENT]
AUTHORIZATION [dbo]
GO
ALTER ROLE [STUDENT] ADD MEMBER [duplicate@example.com]
GO
ALTER ROLE [STUDENT] ADD MEMBER [duplicate8@example.com]
GO
ALTER ROLE [STUDENT] ADD MEMBER [EBTIHAL.DOE@example.com]
GO
ALTER ROLE [STUDENT] ADD MEMBER [invalid3.dept@example.com]
GO
```

```
ALTER ROLE [STUDENT] ADD MEMBER [MUATEF30@example.com]
GO
```

Uses

duplicate@example.com
duplicate8@example.com
EBTIHAL.DOE@example.com
invalid3.dept@example.com
MUATEF30@example.com

Used By

[dbo].[InsertStudentCourse]
[dbo].[sp_add_person]
[dbo].[sp_get_all_students]
[dbo].[sp_get_student_byID]
[dbo].[sp_std_info_depart]

Schemas

Objects

Name
admin_sc
instructor
proced
Student

admin_sc

Properties

Property	Value
Owner	ADMIN

SQL Script

```
CREATE SCHEMA [admin_sc]
AUTHORIZATION [ADMIN]
GO
```

instructor

Properties

Property	Value
Owner	INSTRUCTOR

SQL Script

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

Used By

[instructor].[GET_PERSON]

proced

Properties

Property	Value
Owner	dbo

SQL Script

```
CREATE SCHEMA [proced]
AUTHORIZATION [dbo]
GO
```

Student

Properties

Property	Value
Owner	STUDENT

SQL Script

```
CREATE SCHEMA [Student]
AUTHORIZATION [STUDENT]
GO
```