Examination system

Server	.
Author	user
Created	Wednesday, January 17, 2024 9:25:24 AM
File Path	E:\ITI-.NET-MATERIAL-2023\DB\project\backup script\General script-2024-01-17T09-25-24.pdf

Table of Contents

.

Databases (1)

- Examination system

Server Properties

Property	Value
Product	Microsoft SQL Server
Version	16.0.1110.1
Language	English (United States)
Platform	NT x64
Edition	Developer Edition (64-bit)
Engine Edition	3 (Enterprise)
Processors	8
OS Version	6.3 (22621)
Physical Memory	8033
Is Clustered	False
Root Directory	C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL
Collation	SQL_Latin1_General_CP1_CI_AS

Server Settings

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

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	8
Min server memory	16
Max server memory	2147483647
Scan for startup procs	False
Transform noise words	False
CLR enabled	False
Blocked process threshold	0
Filestream access level	False
Optimize for ad hoc workloads	False
CLR strict security	True

User databases

Databases (1)

- Examination system

Examination system Database

Files

Name	Type	File Group	Size	Maxsize	Autogrowth	File Name
Examination system	Data		72.00 MB	unlimited	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\Examination system.mdf
Examination system_log	Log		8.00 MB	2048.00 GB	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\Examination system_log.ldf
filegroup1	Data	File-Gr1	8.00 MB	unlimited	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\filegroup1.ndf
filegroup2	Data	File-Gr2	8.00 MB	unlimited	64.00 MB	C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\filegroup2.ndf

Tables

Objects

Name
dbo.Branch
dbo.Class
dbo.Class_Course_Instractor
dbo.Course
dbo.Department
dbo.Department_Branch
dbo.Exam
dbo.Exam_Questions
dbo.Exam_Questions_Student
dbo.Instructor
dbo.Intake
dbo.Intake_Instructor
dbo.Question
dbo.Student
dbo.Student_Course
dbo.Student_Exam
dbo.Track
dbo.Track_Course

[dbo].[Branch]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	1
Created	6:04:52 AM Tuesday, January 16, 2024
Last Modified	6:04:53 AM Tuesday, January 16, 2024

Columns

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

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Branch	Branch_id	True
	Branch_unique_phone	Branch_phone	True

Permissions

Type	Action	Owning Principal
Grant	INSERT	training_manager_role
Grant	UPDATE	training_manager_role

SQL Script

```
CREATE TABLE [dbo].[Branch]
(
[Branch_id] [int] NOT NULL IDENTITY(1, 1),
[Branch_name] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Branch_address] [nvarchar] (max) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Branch_phone] [int] NULL
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[Branch] ADD CONSTRAINT [PK_Branch] PRIMARY KEY CLUSTERED ([Branch_id]) ON
[PRIMARY]
GO
ALTER TABLE [dbo].[Branch] ADD CONSTRAINT [Branch_unique_phone] UNIQUE NONCLUSTERED
([Branch_phone]) ON [PRIMARY]
GO
GRANT INSERT ON  [dbo].[Branch] TO [training_manager_role]
GO
GRANT UPDATE ON  [dbo].[Branch] TO [training_manager_role]
GO
```

Used By

[dbo].[Class]
[dbo].[Department_Branch]
[dbo].[Intake]
[dbo].[HRBranch]

[dbo].[Class]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	2
Created	3:27:54 AM Tuesday, January 16, 2024
Last Modified	6:05:31 PM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Class_id	int	4	NOT NULL	1 - 1
	Class_name	nvarchar(50)	100	NOT NULL	
	Class_Floor	int	4	NOT NULL	
FK	Track_Id	int	4	NOT NULL	
FK	Dept_id	int	4	NOT NULL	
FK	Branch_id	int	4	NOT NULL	

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Class	Class_id	True
	Class_unique	Branch_id, Dept_id, Track_Id	True
	Class_name_index	Class_name	

Foreign Keys

Name	Columns
Class_Branch_FK	Branch_id->[dbo].[Branch].[Branch_id]
Class_Department_FK	Dept_id->[dbo].[Department].[Dept_id]
Class_Track_FK	Track_Id->[dbo].[Track].[Track_id]

SQL Script

```
CREATE TABLE [dbo].[Class]
(
[Class_id] [int] NOT NULL IDENTITY(1, 1),
[Class_name] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Class_Floor] [int] NOT NULL,
[Track_Id] [int] NOT NULL,
[Dept_id] [int] NOT NULL,
[Branch_id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Class] ADD CONSTRAINT [PK_Class] PRIMARY KEY CLUSTERED ([Class_id]) ON
[PRIMARY]
GO
ALTER TABLE [dbo].[Class] ADD CONSTRAINT [Class_unique] UNIQUE NONCLUSTERED ([Branch_id],
[Dept_id], [Track_Id]) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [Class_name_index] ON [dbo].[Class] ([Class_name]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Class] ADD CONSTRAINT [Class_Branch_FK] FOREIGN KEY ([Branch_id]) REFERENCES
[dbo].[Branch] ([Branch_id])
GO
ALTER TABLE [dbo].[Class] ADD CONSTRAINT [Class_Department_FK] FOREIGN KEY ([Dept_id]) REFERENCES
[dbo].[Department] ([Dept_id])
GO
ALTER TABLE [dbo].[Class] ADD CONSTRAINT [Class_Track_FK] FOREIGN KEY ([Track_Id]) REFERENCES
[dbo].[Track] ([Track_id])
GO
```

Uses

[dbo].[Branch]
[dbo].[Department]
[dbo].[Track]

Used By

[dbo].[Class_Course_Instractor]
[dbo].[Student]
[dbo].[HRClass]

[dbo].[Class_Course_Instractor]

Properties

Property	Value
Row Count (~)	3
Created	4:56:46 AM Tuesday, January 16, 2024
Last Modified	6:49:19 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK	Class_Id	int	4	NOT NULL
PK FK FK	Course_Id	int	4	NOT NULL
PK	Year	int	4	NOT NULL
	Instractor_ID	int	4	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK	PK_Class_Course_Instractor	Class_Id, Course_Id, Year	True

Foreign Keys

Name	Columns
Class_Course_Instractor_class_FK	Class_Id->[dbo].[Class].[Class_id]
Class_Course_Instractor_Course_FK	Course_Id->[dbo].[Course].[Crs_id]
Class_Course_Instractor_Instructor_FK	Course_Id->[dbo].[Instructor].[Ins_Id]

SQL Script

```
CREATE TABLE [dbo].[Class_Course_Instractor]
(
[Class_Id] [int] NOT NULL,
[Course_Id] [int] NOT NULL,
[Year] [int] NOT NULL,
[Instractor_ID] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Class_Course_Instractor] ADD CONSTRAINT [PK_Class_Course_Instractor] PRIMARY
```

```
KEY CLUSTERED ([Class_Id], [Course_Id], [Year]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Class_Course_Instractor] ADD CONSTRAINT [Class_Course_Instractor_class_FK]
FOREIGN KEY ([Class_Id]) REFERENCES [dbo].[Class] ([Class_id])
GO
ALTER TABLE [dbo].[Class_Course_Instractor] ADD CONSTRAINT [Class_Course_Instractor_Course_FK]
FOREIGN KEY ([Course_Id]) REFERENCES [dbo].[Course] ([Crs_id])
GO
ALTER TABLE [dbo].[Class_Course_Instractor] ADD CONSTRAINT [Class_Course_Instractor_Instructor_-
FK] FOREIGN KEY ([Course_Id]) REFERENCES [dbo].[Instructor] ([Ins_Id])
GO
```

Uses

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

Used By

[dbo].[HR_Class_Course_Instructor]
[dbo].[HRClassCourseInstructor]

[dbo].[Course]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	10
Created	3:36:04 AM Tuesday, January 16, 2024
Last Modified	8:14:52 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK	Crs_id	int	4	NOT NULL	1 - 1
	Crs_name	nvarchar(50)	100	NOT NULL	
	Crs_description	nvarchar(max)	max	NULL allowed	
	Crs_minDegree	int	4	NOT NULL	
	Crs_maxDegree	int	4	NOT NULL	

Indexes

Key	Name	Key Columns	Unique
PK	PK_Courses	Crs_id	True
	Course_name_index	Crs_name	

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
InsertTriggerMessage	True	True	After Insert
InsertTriggerMessageCourse	True	True	After Insert

Check Constraints

Name	On Column	Constraint
Course_check_maxDegree	Crs_maxDegree	([Crs_maxDegree]>=(10) AND [Crs_maxDegree]<=(100))
Course_check_minDegree	Crs_minDegree	([Crs_minDegree]>=(0) AND [Crs_minDegree]<=(50))

Permissions

Type	Action	Owning Principal
Grant	DELETE	training_manager_role
Grant	INSERT	training_manager_role
Grant	SELECT	training_manager_role
Grant	UPDATE	training_manager_role

SQL Script

```
CREATE TABLE [dbo].[Course]
(
[Crs_id] [int] NOT NULL IDENTITY(1, 1),
[Crs_name] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Crs_description] [nvarchar] (max) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
[Crs_minDegree] [int] NOT NULL,
[Crs_maxDegree] [int] NOT NULL
) ON [PRIMARY]
GO
create trigger [dbo].[InsertTriggerMessage]
on [dbo].[Course]
after insert
as
    select 'Added Done'
GO
create   trigger [dbo].[InsertTriggerMessageCourse]
on [dbo].[Course]
after insert
as
    select 'Added Done'

GO
ALTER TABLE [dbo].[Course] ADD CONSTRAINT [Course_check_maxDegree] CHECK (([Crs_maxDegree]>=(10)
AND [Crs_maxDegree]<=(100)))
GO
ALTER TABLE [dbo].[Course] ADD CONSTRAINT [Course_check_minDegree] CHECK (([Crs_minDegree]>=(0)
AND [Crs_minDegree]<=(50)))
GO
ALTER TABLE [dbo].[Course] ADD CONSTRAINT [PK_Courses] PRIMARY KEY CLUSTERED ([Crs_id]) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [Course_name_index] ON [dbo].[Course] ([Crs_name]) ON [PRIMARY]
GO
GRANT DELETE ON  [dbo].[Course] TO [training_manager_role]
GO
GRANT INSERT ON  [dbo].[Course] TO [training_manager_role]
GO
GRANT SELECT ON  [dbo].[Course] TO [training_manager_role]
GO
GRANT UPDATE ON  [dbo].[Course] TO [training_manager_role]
GO
```

Used By

[dbo].[Class_Course_Instractor]
[dbo].[Exam]
[dbo].[Question]
[dbo].[Student_Course]
[dbo].[Track_Course]
[dbo].[GetTotalDegreeForStudentExam]
[dbo].[GetTotalDegreeForStudentExamAndStatus]
[dbo].[HRCousres]

[dbo].[Department]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	4
Created	3:25:28 AM Tuesday, January 16, 2024
Last Modified	4:43:36 AM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Dept_id	int	4	NOT NULL	1 - 1
	Dept_name	nvarchar(50)	100	NOT NULL	
	Dept_phone	int	4	NULL allowed	

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Department	Dept_id	True
	Department_unique_phone	Dept_phone	True

SQL Script

```
CREATE TABLE [dbo].[Department]
(
[Dept_id] [int] NOT NULL IDENTITY(1, 1),
[Dept_name] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Dept_phone] [int] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Department] ADD CONSTRAINT [PK_Department] PRIMARY KEY CLUSTERED ([Dept_id])
ON [PRIMARY]
GO
ALTER TABLE [dbo].[Department] ADD CONSTRAINT [Department_unique_phone] UNIQUE NONCLUSTERED
([Dept_phone]) ON [PRIMARY]
GO
```

Used By

[dbo].[Class]
[dbo].[Department_Branch]
[dbo].[Track]
[dbo].[HRDepartment]

[dbo].[Department_Branch]

Properties

Property	Value
Row Count (~)	4
Created	7:44:17 PM Monday, January 15, 2024
Last Modified	6:04:53 AM Tuesday, January 16, 2024

Columns

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

Indexes

Key	Name	Key Columns	Unique
PK	PK_Department_Branch	Dept_id, Branch_id	True

Foreign Keys

Name	Columns
Department_Branch_Branch_FK	Branch_id->[dbo].[Branch].[Branch_id]
Department_Branch_Dept_FK	Dept_id->[dbo].[Department].[Dept_id]

SQL Script

```
CREATE TABLE [dbo].[Department_Branch]
(
[Dept_id] [int] NOT NULL,
[Branch_id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Department_Branch] ADD CONSTRAINT [PK_Department_Branch] PRIMARY KEY CLUSTERED
([Dept_id], [Branch_id]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Department_Branch] ADD CONSTRAINT [Department_Branch_Branch_FK] FOREIGN KEY
([Branch_id]) REFERENCES [dbo].[Branch] ([Branch_id])
GO
ALTER TABLE [dbo].[Department_Branch] ADD CONSTRAINT [Department_Branch_Dept_FK] FOREIGN KEY
```

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

Uses

[dbo].[Branch]
[dbo].[Department]

Used By

[dbo].[HRDeptBranch]

[dbo].[Exam]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	4
Created	4:32:52 AM Tuesday, January 16, 2024
Last Modified	8:02:31 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Computed	Max Length (Bytes)	Nullability	Identity
🔑	Exam_id	int		4	NOT NULL	1 - 1
🖧	Exam_start	time		5	NOT NULL	
	Exam_end	time		5	NOT NULL	
🖧 ▦	Exam_date	date		3	NOT NULL	
▦	Exam_type	nvarchar(50)		100	NOT NULL	
🖧 🔑	Course_id	int		4	NOT NULL	
🔑	Instructor_id	int		4	NOT NULL	
🖧 🔑	Intake_id	int		4	NOT NULL	
	Total_time	int	True	4	NULL allowed	

Computed columns

Name	Column definition
Total_time	(datediff(minute,[Exam_start],[Exam_end]))

Indexes

Key	Name	Key Columns	Unique
🔑	PK_Exam	Exam_id	True
	Exam_unique	Exam_date, Exam_start, Course_id, Intake_id	True

Check Constraints

Name	On Column	Constraint

Exam_check_Type	Exam_type	([Exam_type]='Normal' OR [Exam_type]='Corrective')
Exam_check_date	Exam_date	(datepart(day,CONVERT([date],[Exam_date],(0)))>=datepart(day,getdate()))
Exam_check_end		([Exam_end]<>[Exam_start] AND [Exam_end]>[Exam_start])
Exam_check_start		([Exam_start]<>[Exam_end] AND [Exam_start]<[Exam_end])

Foreign Keys

Name	Columns
Exam_course_FK	Course_id->[dbo].[Course].[Crs_id]
Exam_Instructor_FK	Instructor_id->[dbo].[Instructor].[Ins_Id]
Exam_Intake_FK	Intake_id->[dbo].[Intake].[Int_id]

Permissions

Type	Action	Owning Principal
Grant	INSERT	instructor_role
Grant	SELECT	Student
Grant	SELECT	student_role

SQL Script

```
CREATE TABLE [dbo].[Exam]
(
[Exam_id] [int] NOT NULL IDENTITY(1, 1),
[Exam_start] [time] NOT NULL,
[Exam_end] [time] NOT NULL,
[Exam_date] [date] NOT NULL,
[Exam_type] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Course_id] [int] NOT NULL,
[Instructor_id] [int] NOT NULL,
[Intake_id] [int] NOT NULL,
[Total_time] AS (datediff(minute,[Exam_start],[Exam_end]))
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_check_Type] CHECK (([Exam_type]='Normal' OR
[Exam_type]='Corrective'))
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_check_date] CHECK
((datepart(day,CONVERT([date],[Exam_date],(0)))>=datepart(day,getdate())))
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_check_end] CHECK (([Exam_end]<>[Exam_start] AND
[Exam_end]>[Exam_start]))
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_check_start] CHECK (([Exam_start]<>[Exam_end] AND
[Exam_start]<[Exam_end]))
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [PK_Exam] PRIMARY KEY CLUSTERED ([Exam_id]) ON [PRIMARY]
```

```
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_unique] UNIQUE NONCLUSTERED ([Exam_date],
[Exam_start], [Course_id], [Intake_id]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_course_FK] FOREIGN KEY ([Course_id]) REFERENCES
[dbo].[Course] ([Crs_id])
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_Instructor_FK] FOREIGN KEY ([Instructor_id])
REFERENCES [dbo].[Instructor] ([Ins_Id])
GO
ALTER TABLE [dbo].[Exam] ADD CONSTRAINT [Exam_Intake_FK] FOREIGN KEY ([Intake_id]) REFERENCES
[dbo].[Intake] ([Int_id])
GO
GRANT INSERT ON  [dbo].[Exam] TO [instructor_role]
GO
GRANT SELECT ON  [dbo].[Exam] TO [Student]
GO
GRANT SELECT ON  [dbo].[Exam] TO [student_role]
GO
```

Uses

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

Used By

[dbo].[Exam_Questions]
[dbo].[Exam_Questions_Student]
[dbo].[Student_Exam]
[dbo].[DeleteExamProc]
[dbo].[GetDetailsForExam]
[dbo].[GetExamsByCourse]
[dbo].[GetExamsByDate]
[dbo].[GetExamsByDateRange]
[dbo].[GetTotalDegreeForStudentExam]
[dbo].[GetTotalDegreeForStudentExamAndStatus]
[dbo].[GetStudentTotalDegreeFromHisAnswers]
[dbo].[HRExam]

[dbo].[Exam_Questions]

Properties

Property	Value
Row Count (~)	20
Created	5:27:22 AM Wednesday, January 17, 2024
Last Modified	5:27:22 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK	Exam_Id	int	4	NOT NULL
PK FK	Question_Id	int	4	NOT NULL
	Degree	float	8	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK	PK_Exam_Questions	Exam_Id, Question_Id	True

Check Constraints

Name	On Column	Constraint
Exam_Questions_check_degree	Degree	([Degree]>=(1) AND [Degree]<=(10))

Foreign Keys

Name	Columns
Exam_Questions_Exam_FK	Exam_Id->[dbo].[Exam].[Exam_id]
Exam_Questions_Question_FK	Question_Id->[dbo].[Question].[Question_id]

Permissions

Type	Action	Owning Principal
Grant	INSERT	instructor_role

SQL Script

```
CREATE TABLE [dbo].[Exam_Questions]
(
[Exam_Id] [int] NOT NULL,
[Question_Id] [int] NOT NULL,
[Degree] [float] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam_Questions] ADD CONSTRAINT [Exam_Questions_check_degree] CHECK
(([Degree]>=(1) AND [Degree]<=(10)))
GO
ALTER TABLE [dbo].[Exam_Questions] ADD CONSTRAINT [PK_Exam_Questions] PRIMARY KEY CLUSTERED
([Exam_Id], [Question_Id]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam_Questions] ADD CONSTRAINT [Exam_Questions_Exam_FK] FOREIGN KEY ([Exam_-
Id]) REFERENCES [dbo].[Exam] ([Exam_id])
GO
ALTER TABLE [dbo].[Exam_Questions] ADD CONSTRAINT [Exam_Questions_Question_FK] FOREIGN KEY
([Question_Id]) REFERENCES [dbo].[Question] ([Question_id])
GO
GRANT INSERT ON  [dbo].[Exam_Questions] TO [instructor_role]
GO
```

Uses

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

Used By

[dbo].[HR_Exam_Questions]

[dbo].[Exam_Questions_Student]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	10
Created	7:49:47 PM Monday, January 15, 2024
Last Modified	5:53:33 AM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK	Exam_Id	int	4	NOT NULL
PK FK	Question_Id	int	4	NOT NULL
PK FK	Student_Id	int	4	NOT NULL
	Student_answer	nvarchar(50)	100	NOT NULL

Indexes

Key	Name	Key Columns	Unique
PK	PK_Exam_Questions_Student	Exam_Id, Question_Id, Student_Id	True

Foreign Keys

Name	Columns
Exam_Questions_Student_Exam_FK	Exam_Id->[dbo].[Exam].[Exam_id]
Exam_Questions_Student_Quest_FK	Question_Id->[dbo].[Question].[Question_id]
Exam_Questions_Student_Std_FK	Student_Id->[dbo].[Student].[Std_id]

SQL Script

```
CREATE TABLE [dbo].[Exam_Questions_Student]
(
[Exam_Id] [int] NOT NULL,
[Question_Id] [int] NOT NULL,
[Student_Id] [int] NOT NULL,
[Student_answer] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
```

```
GO
ALTER TABLE [dbo].[Exam_Questions_Student] ADD CONSTRAINT [PK_Exam_Questions_Student] PRIMARY KEY
CLUSTERED ([Exam_Id], [Question_Id], [Student_Id]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Exam_Questions_Student] ADD CONSTRAINT [Exam_Questions_Student_Exam_FK]
FOREIGN KEY ([Exam_Id]) REFERENCES [dbo].[Exam] ([Exam_id])
GO
ALTER TABLE [dbo].[Exam_Questions_Student] ADD CONSTRAINT [Exam_Questions_Student_Quest_FK]
FOREIGN KEY ([Question_Id]) REFERENCES [dbo].[Question] ([Question_id])
GO
ALTER TABLE [dbo].[Exam_Questions_Student] ADD CONSTRAINT [Exam_Questions_Student_Std_FK] FOREIGN
KEY ([Student_Id]) REFERENCES [dbo].[Student] ([Std_id])
GO
```

Uses

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

Used By

[dbo].[GetExamResultForStudent]
[dbo].[HR_Exam_Questions_Student]

[dbo].[Instructor]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	6
Created	3:56:00 AM Tuesday, January 16, 2024
Last Modified	7:23:50 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
	Ins_Id	int	4	NOT NULL	1 - 1
	Ins_Name	nvarchar(50)	100	NOT NULL	
	Ins_Age	int	4	NULL allowed	
	Ins_Address	nvarchar(50)	100	NULL allowed	
	Ins_Phone	int	4	NULL allowed	

Indexes

Key	Name	Key Columns	Unique
	PK_Instructor	Ins_Id	True
	Instructor_unique_phone	Ins_Phone	True
	Instructor_name_index	Ins_Name	

Triggers

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

Check Constraints

Name	On Column	Constraint
Instructor_check_age	Ins_Age	([Ins_age]>=(30) AND [Ins_age]<=(60))

Permissions

Type	Action	Owning Principal
Grant	DELETE	training_manager_role
Grant	INSERT	training_manager_role
Grant	SELECT	training_manager_role
Grant	UPDATE	training_manager_role

SQL Script

```
CREATE TABLE [dbo].[Instructor]
(
[Ins_Id] [int] NOT NULL IDENTITY(1, 1),
[Ins_Name] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Ins_Age] [int] NULL,
[Ins_Address] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
[Ins_Phone] [int] NULL
) ON [PRIMARY]
GO
create trigger [dbo].[InsertTriggerMessageInstructor]
on [dbo].[Instructor]
after insert
as
    select 'Added Done'
GO
ALTER TABLE [dbo].[Instructor] ADD CONSTRAINT [Instructor_check_age] CHECK (([Ins_age]>=(30) AND
[Ins_age]<=(60)))
GO
ALTER TABLE [dbo].[Instructor] ADD CONSTRAINT [PK_Instructor] PRIMARY KEY CLUSTERED ([Ins_Id]) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [Instructor_name_index] ON [dbo].[Instructor] ([Ins_Name]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Instructor] ADD CONSTRAINT [Instructor_unique_phone] UNIQUE NONCLUSTERED
([Ins_Phone]) ON [PRIMARY]
GO
GRANT DELETE ON  [dbo].[Instructor] TO [training_manager_role]
GO
GRANT INSERT ON  [dbo].[Instructor] TO [training_manager_role]
GO
GRANT SELECT ON  [dbo].[Instructor] TO [training_manager_role]
GO
GRANT UPDATE ON  [dbo].[Instructor] TO [training_manager_role]
GO
```

Used By

[dbo].[Class_Course_Instractor]
[dbo].[Exam]

[dbo].[Intake_Instructor]
[dbo].[Question]
[dbo].[HRInstructoe]

[dbo].[Intake]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	2
Created	3:27:18 AM Tuesday, January 16, 2024
Last Modified	7:17:56 PM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
PK C	Int_id	int	4	NOT NULL	1 - 1
	Int_name	nvarchar(50)	100	NOT NULL	
FK	Branch_id	int	4	NOT NULL	

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Intake	Int_id	True
	Intake_name_index	Int_name	

Foreign Keys

Name	Columns
Intake_Branch_FK	Branch_id->[dbo].[Branch].[Branch_id]

Permissions

Type	Action	Owning Principal
Grant	INSERT	training_manager_role
Grant	UPDATE	training_manager_role

SQL Script

```
CREATE TABLE [dbo].[Intake]
(
[Int_id] [int] NOT NULL IDENTITY(1, 1),
[Int_name] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Branch_id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Intake] ADD CONSTRAINT [PK_Intake] PRIMARY KEY CLUSTERED ([Int_id]) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [Intake_name_index] ON [dbo].[Intake] ([Int_name]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Intake] ADD CONSTRAINT [Intake_Branch_FK] FOREIGN KEY ([Branch_id]) REFERENCES
[dbo].[Branch] ([Branch_id])
GO
GRANT INSERT ON  [dbo].[Intake] TO [training_manager_role]
GO
GRANT UPDATE ON  [dbo].[Intake] TO [training_manager_role]
GO
```

Uses

[dbo].[Branch]

Used By

[dbo].[Exam]
[dbo].[Intake_Instructor]
[dbo].[Student]
[dbo].[HRIntake]

[dbo].[Intake_Instructor]

Properties

Property	Value
Row Count (~)	5
Created	5:58:56 AM Tuesday, January 16, 2024
Last Modified	5:59:04 AM Tuesday, January 16, 2024

Columns

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

Indexes

Key	Name	Key Columns	Unique
PK	PK_Intake_Instructor	Intake_Id, Ins_Id	True

Foreign Keys

Name	Columns
Intake_Instructor_Ins_FK	Ins_Id->[dbo].[Instructor].[Ins_Id]
Intake_Instructor_Intake_FK	Intake_Id->[dbo].[Intake].[Int_id]

SQL Script

```
CREATE TABLE [dbo].[Intake_Instructor]
(
[Ins_Id] [int] NOT NULL,
[Intake_Id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Intake_Instructor] ADD CONSTRAINT [PK_Intake_Instructor] PRIMARY KEY CLUSTERED
([Intake_Id], [Ins_Id]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Intake_Instructor] ADD CONSTRAINT [Intake_Instructor_Ins_FK] FOREIGN KEY
([Ins_Id]) REFERENCES [dbo].[Instructor] ([Ins_Id])
GO
ALTER TABLE [dbo].[Intake_Instructor] ADD CONSTRAINT [Intake_Instructor_Intake_FK] FOREIGN KEY
```

```
([Intake_Id]) REFERENCES [dbo].[Intake] ([Int_id])
GO
```

Uses

[dbo].[Instructor]
[dbo].[Intake]

Used By

[dbo].[HRInstrucot_Intake]

[dbo].[Question]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	14
Created	3:56:32 AM Tuesday, January 16, 2024
Last Modified	5:27:22 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
	Type	nvarchar(50)	100	NOT NULL	
	QuestionTitle	nvarchar(max)	max	NOT NULL	
	Correct_answer	nvarchar(50)	100	NOT NULL	
FK	Course_id	int	4	NOT NULL	
FK C	Question_id	int	4	NOT NULL	1 - 1
FK	Instructor_id	int	4	NOT NULL	

Indexes

Key	Name	Key Columns	Unique
FK C	PK_Question	Question_id	True

Check Constraints

Name	On Column	Constraint
Question_check_Type	Type	([Type]='MCQ' OR [type]='BOOLEAN' OR [type]='TEXT')

Foreign Keys

Name	Columns
Question_Course_FK	Course_id->[dbo].[Course].[Crs_id]
Question_Instructor_FK	Instructor_id->[dbo].[Instructor].[Ins_Id]

Permissions

Type	Action	Owning Principal
Grant	DELETE	instructor_role
Grant	INSERT	instructor_role
Grant	SELECT	student_role
Grant	UPDATE	instructor_role

SQL Script

```
CREATE TABLE [dbo].[Question]
(
[Type] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[QuestionTitle] [nvarchar] (max) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Correct_answer] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Course_id] [int] NOT NULL,
[Question_id] [int] NOT NULL IDENTITY(1, 1),
[Instructor_id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [Question_check_Type] CHECK (([Type]='MCQ' OR
[type]='BOOLEAN' OR [type]='TEXT'))
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [PK_Question] PRIMARY KEY CLUSTERED ([Question_id])
ON [PRIMARY]
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [Question_Course_FK] FOREIGN KEY ([Course_id])
REFERENCES [dbo].[Course] ([Crs_id])
GO
ALTER TABLE [dbo].[Question] ADD CONSTRAINT [Question_Instructor_FK] FOREIGN KEY
([Instructor_id]) REFERENCES [dbo].[Instructor] ([Ins_Id])
GO
GRANT DELETE ON  [dbo].[Question] TO [instructor_role]
GO
GRANT INSERT ON  [dbo].[Question] TO [instructor_role]
GO
GRANT UPDATE ON  [dbo].[Question] TO [instructor_role]
GO
GRANT SELECT ON  [dbo].[Question] TO [student_role]
GO
```

Uses

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

Used By

[dbo].[Exam_Questions]
[dbo].[Exam_Questions_Student]

[dbo].[GetExamResultForStudent]
[dbo].[getQuestionsForSpecificCourse]
[dbo].[HRQuestions]

[dbo].[Student]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	5
Created	3:29:41 AM Tuesday, January 16, 2024
Last Modified	8:14:52 AM Wednesday, January 17, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability	Identity
	Std_id	int	4	NOT NULL	1 - 1
	Std_name	nvarchar(50)	100	NOT NULL	
	Std_age	int	4	NOT NULL	
	Std_address	nvarchar(50)	100	NULL allowed	
	Std_phone	int	4	NULL allowed	
	Intake_id	int	4	NOT NULL	
	Class_id	int	4	NOT NULL	

Indexes

Key	Name	Key Columns	Unique
	PK_Student	Std_id	True
	Student_unique_phone	Std_phone	True
	Student_name_index	Std_name	

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
PreventStudentUpdate_TRIGGER	True	True	Instead Of Update

Check Constraints

Name	On Column	Constraint
Student_check_age	Std_age	([Std_Age]>=(10) AND [Std_Age]<=(100))

Foreign Keys

Name	Columns
Student_Class_FK	Class_id->[dbo].[Class].[Class_id]
Student_Intake_FK	Intake_id->[dbo].[Intake].[Int_id]

Permissions

Type	Action	Owning Principal
Grant	INSERT	training_manager_role
Grant	UPDATE	training_manager_role

SQL Script

```
CREATE TABLE [dbo].[Student]
(
[Std_id] [int] NOT NULL IDENTITY(1, 1),
[Std_name] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Std_age] [int] NOT NULL,
[Std_address] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
[Std_phone] [int] NULL,
[Intake_id] [int] NOT NULL,
[Class_id] [int] NOT NULL
) ON [PRIMARY]
GO
--Prevent Student to update if has relation with courses
CREATE  TRIGGER [dbo].[PreventStudentUpdate_TRIGGER]
ON [dbo].[Student]
INSTEAD OF UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT *
        FROM inserted i
        INNER JOIN HRStudent_Course sc ON i.Std_id = sc.Student_id
    )
    BEGIN
        RAISERROR ('Cannot update Student Becouse have Relationship with courses', 16, 1);
    END
    ELSE
    BEGIN
        UPDATE s
        SET s.Std_name = i.Std_name, s.Std_age = i.Std_age, s.Std_address = i.Std_address,
s.Intake_id = i.Intake_id , s.Class_id = i.Class_id
        FROM Student s
        INNER JOIN inserted i ON s.Std_id = i.Std_id;
    END
```

```
END;

--update Student
--set Std_name = 'omar' , Std_age = 22 , Std_address = 'sharqia' , Std_phone = '01004325257' ,
Intake_id = 1 , Class_id = 1
--where Std_id = 5

SELECT DaY(GETDATE())
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [Student_check_age] CHECK (([Std_Age]>=(10) AND [Std_-
Age]<=(100)))
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [PK_Student] PRIMARY KEY CLUSTERED ([Std_id]) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [Student_name_index] ON [dbo].[Student] ([Std_name]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [Student_unique_phone] UNIQUE NONCLUSTERED
([Std_phone]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [Student_Class_FK] FOREIGN KEY ([Class_id]) REFERENCES
[dbo].[Class] ([Class_id])
GO
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [Student_Intake_FK] FOREIGN KEY ([Intake_id])
REFERENCES [dbo].[Intake] ([Int_id])
GO
GRANT INSERT ON  [dbo].[Student] TO [training_manager_role]
GO
GRANT UPDATE ON  [dbo].[Student] TO [training_manager_role]
GO
```

Uses

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

Used By

[dbo].[Exam_Questions_Student]
[dbo].[Student_Course]
[dbo].[Student_Exam]
[dbo].[CreateStudent]
[dbo].[DeleteStudent]
[dbo].[EditStudent]
[dbo].[GetStudentTotalDegreeFromHisAnswers]
[dbo].[HRStudent]

[dbo].[Student_Course]

Properties

Property	Value
Row Count (~)	5
Created	1:05:07 AM Wednesday, January 17, 2024
Last Modified	1:05:07 AM Wednesday, January 17, 2024

Columns

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

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Student_Course	Course_id, Student_id	True

Foreign Keys

Name	Columns
Student_Course_Course_FK	Course_id->[dbo].[Course].[Crs_id]
Student_Course_student_FK	Student_id->[dbo].[Student].[Std_id]

SQL Script

```
CREATE TABLE [dbo].[Student_Course]
(
[Student_id] [int] NOT NULL,
[Course_id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Course] ADD CONSTRAINT [PK_Student_Course] PRIMARY KEY CLUSTERED
([Course_id], [Student_id]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Course] ADD CONSTRAINT [Student_Course_Course_FK] FOREIGN KEY
([Course_id]) REFERENCES [dbo].[Course] ([Crs_id])
GO
ALTER TABLE [dbo].[Student_Course] ADD CONSTRAINT [Student_Course_student_FK] FOREIGN KEY
```

```
([Student_id]) REFERENCES [dbo].[Student] ([Std_id])
GO
```

Uses

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

Used By

[dbo].[HRStudent_Course]

[dbo].[Student_Exam]

Properties

Property	Value
Row Count (~)	6
Created	7:47:13 PM Monday, January 15, 2024
Last Modified	8:46:32 PM Tuesday, January 16, 2024

Columns

Key	Name	Data Type	Max Length (Bytes)	Nullability
PK FK	Student_id	int	4	NOT NULL
PK FK	Exam_id	int	4	NOT NULL
	Exam_result	int	4	NULL allowed

Indexes

Key	Name	Key Columns	Unique
PK	PK_Student_Exam	Student_id, Exam_id	True

Foreign Keys

Name	Columns
Student_Exam_Exam_FK	Exam_id->[dbo].[Exam].[Exam_id]
Student_Exam_Std_FK	Student_id->[dbo].[Student].[Std_id]

Permissions

Type	Action	Owning Principal
Grant	INSERT	instructor_role

SQL Script

```
CREATE TABLE [dbo].[Student_Exam]
(
[Student_id] [int] NOT NULL,
[Exam_id] [int] NOT NULL,
[Exam_result] [int] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Exam] ADD CONSTRAINT [PK_Student_Exam] PRIMARY KEY CLUSTERED
([Student_id], [Exam_id]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Student_Exam] ADD CONSTRAINT [Student_Exam_Exam_FK] FOREIGN KEY ([Exam_id])
REFERENCES [dbo].[Exam] ([Exam_id])
GO
ALTER TABLE [dbo].[Student_Exam] ADD CONSTRAINT [Student_Exam_Std_FK] FOREIGN KEY ([Student_id])
REFERENCES [dbo].[Student] ([Std_id])
GO
GRANT INSERT ON  [dbo].[Student_Exam] TO [instructor_role]
GO
```

Uses

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

Used By

[dbo].[HR_Student_Exam]

[dbo].[Track]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
Row Count (~)	2
Created	3:26:32 AM Tuesday, January 16, 2024
Last Modified	5:57:11 PM Tuesday, January 16, 2024

Columns

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

Indexes

Key	Name	Key Columns	Unique
PK C	PK_Track	Track_id	True
	Track_name_index	Track_name	

Foreign Keys

Name	Columns
Track_Department_FK	Dept_id->[dbo].[Department].[Dept_id]

Permissions

Type	Action	Owning Principal
Grant	INSERT	training_manager_role
Grant	UPDATE	training_manager_role

SQL Script

```
CREATE TABLE [dbo].[Track]
(
[Track_id] [int] NOT NULL IDENTITY(1, 1),
[Track_name] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Dept_id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Track] ADD CONSTRAINT [PK_Track] PRIMARY KEY CLUSTERED ([Track_id]) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [Track_name_index] ON [dbo].[Track] ([Track_name]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Track] ADD CONSTRAINT [Track_Department_FK] FOREIGN KEY ([Dept_id]) REFERENCES
[dbo].[Department] ([Dept_id])
GO
GRANT INSERT ON  [dbo].[Track] TO [training_manager_role]
GO
GRANT UPDATE ON  [dbo].[Track] TO [training_manager_role]
GO
```

Uses

[dbo].[Department]

Used By

[dbo].[Class]
[dbo].[Track_Course]
[dbo].[HRTrack]

[dbo].[Track_Course]

Properties

Property	Value
Row Count (~)	10
Created	7:50:40 PM Monday, January 15, 2024
Last Modified	6:19:03 AM Tuesday, January 16, 2024

Columns

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

Indexes

Key	Name	Key Columns	Unique
PK	PK_Track_Course	Course_id, Track_id	True

Foreign Keys

Name	Columns
Track_Course_Course_FK	Course_id->[dbo].[Course].[Crs_id]
Track_Course_Track_FK	Track_id->[dbo].[Track].[Track_id]

SQL Script

```
CREATE TABLE [dbo].[Track_Course]
(
[Course_id] [int] NOT NULL,
[Track_id] [int] NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Track_Course] ADD CONSTRAINT [PK_Track_Course] PRIMARY KEY CLUSTERED
([Course_id], [Track_id]) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Track_Course] ADD CONSTRAINT [Track_Course_Course_FK] FOREIGN KEY
([Course_id]) REFERENCES [dbo].[Course] ([Crs_id])
GO
ALTER TABLE [dbo].[Track_Course] ADD CONSTRAINT [Track_Course_Track_FK] FOREIGN KEY ([Track_id])
```

```
REFERENCES [dbo].[Track] ([Track_id])
GO
```

Uses

[dbo].[Course]
[dbo].[Track]

Used By

[dbo].[CourseNameTnet]
[dbo].[CursNameTpython]

Views

Objects

Name
dbo.CourseNameTnet
dbo.CursNameTpython
dbo.deptBTI
dbo.InstactoeIntake
dbo.InstructorTeachCourse
dbo.StdWtrWdeptwCI
dbo.StudentEnrollInCourses

[dbo].[CourseNameTnet]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	2:04:55 PM Tuesday, January 16, 2024
Last Modified	2:04:55 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Crs_name	nvarchar(50)	100
Track_name	nvarchar(50)	100

SQL Script

```
create view [dbo].[CourseNameTnet]
as
select c.Crs_name,t.Track_name
from HRCousres c inner join [dbo].[Track_Course] tc
on c.Crs_id =tc.Course_id inner join HRTrack t
on t.Track_id = tc.Track_id and t.Track_id=1

GO
```

Uses

[dbo].[Track_Course]
[dbo].[HRCousres]
[dbo].[HRTrack]

[dbo].[CursNameTpython]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	2:04:48 PM Tuesday, January 16, 2024
Last Modified	2:04:48 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Crs_name	nvarchar(50)	100
Track_name	nvarchar(50)	100

SQL Script

```
create view [dbo].[CursNameTpython]
as
select c.Crs_name,t.Track_name
from HRCousres c inner join [dbo].[Track_Course] tc
on c.Crs_id =tc.Course_id inner join HRTrack t
on t.Track_id = tc.Track_id and t.Track_id=1

GO
```

Uses

[dbo].[Track_Course]
[dbo].[HRCousres]
[dbo].[HRTrack]

[dbo].[deptBTI]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	2:05:24 PM Tuesday, January 16, 2024
Last Modified	2:05:24 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Dept_name	nvarchar(50)	100
Branch_name	nvarchar(50)	100
Int_name	nvarchar(50)	100
Track_name	nvarchar(50)	100

SQL Script

```
create view [dbo].[deptBTI]
 as
select Dept_name, Branch_name ,Int_name,Track_name
 from HRDepartment dt full outer join HRDeptBranch db
 on dt.Dept_id =  db.Dept_id full outer join HRBranch B
 ON B.Branch_id=DB.Branch_id full outer JOIN HRIntake I
 on I.Int_id = B.Branch_id full outer join HRTrack T
 on t.Track_id=dt.Dept_id
GO
```

Uses

[dbo].[HRBranch]
[dbo].[HRDepartment]
[dbo].[HRDeptBranch]
[dbo].[HRIntake]
[dbo].[HRTrack]

[dbo].[InstactoeIntake]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	2:05:35 PM Tuesday, January 16, 2024
Last Modified	2:05:35 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Ins_Name	nvarchar(50)	100
Int_name	nvarchar(50)	100

SQL Script

```
create view [dbo].[InstactoeIntake] as
 select Ins_Name,Int_name
 from HRInstructoe I full outer join HRInstrucot_Intake II
 on i.Ins_Id=ii.Ins_Id inner join HRIntake it
 on it.Int_id=ii.Intake_Id
GO
```

Uses

[dbo].[HRInstrucot_Intake]
[dbo].[HRInstructoe]
[dbo].[HRIntake]

[dbo].[InstructorTeachCourse]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	2:05:07 PM Tuesday, January 16, 2024
Last Modified	2:05:07 PM Tuesday, January 16, 2024

Columns

Name	Data Type	Max Length (Bytes)
Ins_Name	nvarchar(50)	100
Crs_name	nvarchar(50)	100

SQL Script

```
 Create view [dbo].[InstructorTeachCourse] as
 select I.Ins_Name , c.Crs_name
 FROM HRInstructoe I INNER JOIN HRClassCourseInstructor CI
 ON I.Ins_Id= CI.Course_Id INNER Join HRCousres C
 on C.Crs_id=CI.Course_Id
GO
```

Uses

[dbo].[HRClassCourseInstructor]
[dbo].[HRCousres]
[dbo].[HRInstructoe]

[dbo].[StdWtrWdeptwCl]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	2:05:40 PM Tuesday, January 16, 2024
Last Modified	2:05:40 PM Tuesday, January 16, 2024

Columns

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

SQL Script

```
create view [dbo].[StdWtrWdeptwCl]
as
SELECT Std_name, Class_name ,Dept_name,Track_name
FROM HRStudent ST inner JOIN HRClass C
ON ST.Std_id = C.Class_id INNER JOIN HRDepartment D
ON C.Class_id=D.Dept_id inner join HRTrack t
on t.Track_id=c.Class_id
GO
```

Uses

[dbo].[HRClass]
[dbo].[HRDepartment]
[dbo].[HRStudent]
[dbo].[HRTrack]

[dbo].[StudentEnrollInCourses]

Properties

Property	Value
Collation	SQL_Latin1_General_CP1_CI_AS
ANSI Nulls On	True
Quoted Identifier On	True
Created	2:04:41 PM Tuesday, January 16, 2024
Last Modified	2:04:41 PM Tuesday, January 16, 2024

Columns

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

SQL Script

```
create view [dbo].[StudentEnrollInCourses]
as
select s.Std_name , c.Crs_name
from HRCousres c inner join HRStudent_Course stc
on c.Crs_id=stc.Course_id inner join HRStudent s
on s.Std_id=stc.Student_id
GO
```

Uses

[dbo].[HRCousres]
[dbo].[HRStudent]
[dbo].[HRStudent_Course]

Synonyms

Objects

Name
dbo.HR_Class_Course_Instructor
dbo.HR_Exam_Questions
dbo.HR_Exam_Questions_Student
dbo.HR_Student_Exam
dbo.HRBranch
dbo.HRClass
dbo.HRClassCourseInstructor
dbo.HRCousres
dbo.HRDepartment
dbo.HRDeptBranch
dbo.HRExam
dbo.HRInstrucot_Intake
dbo.HRInstructoe
dbo.HRIntake
dbo.HRQuestions
dbo.HRStudent
dbo.HRStudent_Course
dbo.HRTrack

[dbo].[HR_Class_Course_Instructor]

Properties

Property	Value
References	[Class_Course_Instractor]

SQL Script

```
CREATE SYNONYM [dbo].[HR_Class_Course_Instructor] FOR [Class_Course_Instractor]
GO
```

Uses

[dbo].[Class_Course_Instractor]

Used By

[dbo].[CreateExamProc]
[dbo].[UpdateExamProc]

[dbo].[HR_Exam_Questions]

Properties

Property	Value
References	[Exam_Questions]

SQL Script

```
CREATE SYNONYM [dbo].[HR_Exam_Questions] FOR [Exam_Questions]
GO
```

Uses

[dbo].[Exam_Questions]

Used By

[dbo].[AddExamToSpecificStudent]
[dbo].[DeleteAllQuestionFromExam]
[dbo].[GetExamResultForStudent]
[dbo].[SelectQuestionsManualForExam]
[dbo].[SelectQuestionsRandomForExam]
[dbo].[UpdateQuestionsForExam]

[dbo].[HR_Exam_Questions_Student]

Properties

Property	Value
References	[Exam_Questions_Student]

SQL Script

```
CREATE SYNONYM [dbo].[HR_Exam_Questions_Student] FOR [Exam_Questions_Student]
GO
```

Uses

[dbo].[Exam_Questions_Student]

Used By

[dbo].[GetExamResultForStudent]
[dbo].[GetTotalDegreeForStudentExam]
[dbo].[GetTotalDegreeForStudentExamAndStatus]

[dbo].[HR_Student_Exam]

Properties

Property	Value
References	[Student_Exam]

SQL Script

```
CREATE SYNONYM [dbo].[HR_Student_Exam] FOR [Student_Exam]
GO
```

Uses

[dbo].[Student_Exam]

Used By

[dbo].[AddExamToSpecificStudent]
[dbo].[AddExamToStudent]
[dbo].[DeleteExamFromStudent]
[dbo].[GetExamResultForStudent]
[dbo].[GetTotalDegreeForStudentExam]
[dbo].[GetTotalDegreeForStudentExamAndStatus]

[dbo].[HRBranch]

Properties

Property	Value
References	[dbo].[Branch]

SQL Script

```
CREATE SYNONYM [dbo].[HRBranch] FOR [dbo].[Branch]
GO
```

Uses

[dbo].[Branch]

Used By

[dbo].[deptBTI]
[dbo].[AddBranch]
[dbo].[DeleteBranch]
[dbo].[UpdateBranch]

[dbo].[HRClass]

Properties

Property	Value
References	[dbo].[Class]

SQL Script

```
CREATE SYNONYM [dbo].[HRClass] FOR [dbo].[Class]
GO
```

Uses

[dbo].[Class]

Used By

[dbo].[StdWtrWdeptwCl]
[dbo].[AssignCourseToInstructor]
[dbo].[DeleteAssignCourseToInstructor]
[dbo].[UpdateAssignCourseToInstructor]

[dbo].[HRClassCourseInstructor]

Properties

Property	Value
References	[dbo].[Class_Course_Instractor]

SQL Script

```
CREATE SYNONYM [dbo].[HRClassCourseInstructor] FOR [dbo].[Class_Course_Instractor]
GO
```

Uses

[dbo].[Class_Course_Instractor]

Used By

[dbo].[InstructorTeachCourse]
[dbo].[AssignCourseToInstructor]
[dbo].[DeleteAssignCourseToInstructor]
[dbo].[UpdateAssignCourseToInstructor]

[dbo].[HRCousres]

Properties

Property	Value
References	[dbo].[Course]

SQL Script

```
CREATE SYNONYM [dbo].[HRCousres] FOR [dbo].[Course]
GO
```

Uses

[dbo].[Course]

Used By

[dbo].[CourseNameTnet]
[dbo].[CursNameTpython]
[dbo].[InstructorTeachCourse]
[dbo].[StudentEnrollInCourses]
[dbo].[AssignCourseToInstructor]
[dbo].[AssignStudentToCourse]
[dbo].[CreateExamProc]
[dbo].[DeleteAssignCourseToInstructor]
[dbo].[DeleteAssignStudentToCourse]
[dbo].[GetExamResultForStudent]
[dbo].[SelectQuestionsManualForExam]
[dbo].[SelectQuestionsRandomForExam]
[dbo].[UpdateAssignCourseToInstructor]
[dbo].[UpdateExamProc]
[dbo].[UpdateQuestionsForExam]

[dbo].[HRDepartment]

Properties

Property	Value
References	[dbo].[Department]

SQL Script

```
CREATE SYNONYM [dbo].[HRDepartment] FOR [dbo].[Department]
GO
```

Uses

[dbo].[Department]

Used By

[dbo].[deptBTI]
[dbo].[StdWtrWdeptwCl]

[dbo].[HRDeptBranch]

Properties

Property	Value
References	[dbo].[Department_Branch]

SQL Script

```
CREATE SYNONYM [dbo].[HRDeptBranch] FOR [dbo].[Department_Branch]
GO
```

Uses

[dbo].[Department_Branch]

Used By

[dbo].[deptBTI]

[dbo].[HRExam]

Properties

Property	Value
References	[dbo].[Exam]

SQL Script

```
CREATE SYNONYM [dbo].[HRExam] FOR [dbo].[Exam]
GO
```

Uses

[dbo].[Exam]

Used By

[dbo].[AddExamToSpecificStudent]
[dbo].[AddExamToStudent]
[dbo].[CreateExamProc]
[dbo].[DeleteAllQuestionFromExam]
[dbo].[DeleteExamFromStudent]
[dbo].[DeleteExamProc]
[dbo].[GetAllExams]
[dbo].[GetExamResultForStudent]
[dbo].[GetTotalDegreeForStudentExam]
[dbo].[GetTotalDegreeForStudentExamAndStatus]
[dbo].[SelectQuestionsManualForExam]
[dbo].[SelectQuestionsRandomForExam]
[dbo].[UpdateExamProc]
[dbo].[UpdateQuestionsForExam]

[dbo].[HRInstrucot_Intake]

Properties

Property	Value
References	[dbo].[Intake_Instructor]

SQL Script

```
CREATE SYNONYM [dbo].[HRInstrucot_Intake] FOR [dbo].[Intake_Instructor]
GO
```

Uses

[dbo].[Intake_Instructor]

Used By

[dbo].[InstactoeIntake]

[dbo].[HRInstructoe]

Properties

Property	Value
References	[dbo].[Instructor]

SQL Script

```
CREATE SYNONYM [dbo].[HRInstructoe] FOR [dbo].[Instructor]
GO
```

Uses

[dbo].[Instructor]

Used By

[dbo].[InstactoeIntake]
[dbo].[InstructorTeachCourse]
[dbo].[AddInstructor]
[dbo].[AssignCourseToInstructor]
[dbo].[CreateExamProc]
[dbo].[DeleteInstructor]
[dbo].[UpdateAssignCourseToInstructor]
[dbo].[UpdateExamProc]
[dbo].[UpdateInstructor]

[dbo].[HRIntake]

Properties

Property	Value
References	[dbo].[Intake]

SQL Script

```
CREATE SYNONYM [dbo].[HRIntake] FOR [dbo].[Intake]
GO
```

Uses

[dbo].[Intake]

Used By

[dbo].[deptBTI]
[dbo].[InstactoeIntake]
[dbo].[CreateExamProc]
[dbo].[UpdateExamProc]

[dbo].[HRQuestions]

Properties

Property	Value
References	[Question]

SQL Script

```
CREATE SYNONYM [dbo].[HRQuestions] FOR [Question]
GO
```

Uses

[dbo].[Question]

Used By

[dbo].[SelectQuestionsManualForExam]
[dbo].[UpdateQuestionsForExam]

[dbo].[HRStudent]

Properties

Property	Value
References	[dbo].[Student]

SQL Script

```
CREATE SYNONYM [dbo].[HRStudent] FOR [dbo].[Student]
GO
```

Uses

[dbo].[Student]

Used By

[dbo].[StdWtrWdeptwCl]
[dbo].[StudentEnrollInCourses]
[dbo].[AddExamToSpecificStudent]
[dbo].[AddExamToStudent]
[dbo].[AssignStudentToCourse]
[dbo].[DeleteAssignStudentToCourse]
[dbo].[DeleteExamFromStudent]
[dbo].[GetExamResultForStudent]
[dbo].[GetTotalDegreeForStudentExam]
[dbo].[GetTotalDegreeForStudentExamAndStatus]

[dbo].[HRStudent_Course]

Properties

Property	Value
References	[dbo].[Student_Course]

SQL Script

```
CREATE SYNONYM [dbo].[HRStudent_Course] FOR [dbo].[Student_Course]
GO
```

Uses

[dbo].[Student_Course]

Used By

[dbo].[StudentEnrollInCourses]
[dbo].[AddExamToSpecificStudent]
[dbo].[AssignStudentToCourse]
[dbo].[DeleteAssignStudentToCourse]

[dbo].[HRTrack]

Properties

Property	Value
References	[dbo].[Track]

SQL Script

```
CREATE SYNONYM [dbo].[HRTrack] FOR [dbo].[Track]
GO
```

Uses

[dbo].[Track]

Used By

[dbo].[CourseNameTnet]
[dbo].[CursNameTpython]
[dbo].[deptBTI]
[dbo].[StdWtrWdeptwCl]

Stored Procedures

Objects

Name
dbo.AddBranch
dbo.AddExamToSpecificStudent
dbo.AddExamToStudent
dbo.AddInstructor
dbo.AssignCourseToInstructor
dbo.AssignStudentToCourse
dbo.CreateExamProc
dbo.CreateStudent
dbo.DeleteAllQuestionFromExam
dbo.DeleteAssignCourseToInstructor
dbo.DeleteAssignStudentToCourse
dbo.DeleteBranch
dbo.DeleteExamFromStudent
dbo.DeleteExamProc
dbo.DeleteInstructor
dbo.DeleteStudent
dbo.EditStudent
dbo.GetAllExams
dbo.GetDetailsForExam
dbo.GetExamResultForStudent
dbo.GetExamsByCourse
dbo.GetExamsByDate
dbo.GetExamsByDateRange
dbo.GetTotalDegreeForStudentExam
dbo.GetTotalDegreeForStudentExamAndStatus
dbo.SelectQuestionsManualForExam
dbo.SelectQuestionsRandomForExam
dbo.UpdateAssignCourseToInstructor
dbo.UpdateBranch
dbo.UpdateExamProc
dbo.UpdateInstructor
dbo.UpdateQuestionsForExam

[dbo].[AddBranch]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Branch_Name	nvarchar(max)	max
@Branch_Address	nvarchar(max)	max
@Branch_Phone	nvarchar(max)	max

SQL Script

```
create   proc [dbo].[AddBranch]
    @Branch_Name nvarchar(MAX),
     @Branch_Address nvarchar(MAX),
      @Branch_Phone nvarchar(MAX)
as
begin
    BEGIN TRY
        INSERT INTO HRBranch
        VALUES(@Branch_Name , @Branch_Address , @Branch_Phone);
        SELECT 'Branch is Added Successfully'
    END TRY
    BEGIN CATCH
        SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
    END CATCH
end
GO
```

Uses

[dbo].[HRBranch]

[dbo].[AddExamToSpecificStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   proc [dbo].[AddExamToSpecificStudent]
            @Student_id int,
            @Exam_id int


as
BEGIN
    IF Not Exists(select Std_id from HRStudent where Std_id = @student_id) or
        Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id)
        BEGIN
            SELECT 'Something is wrong Exam or Student may be not found try again' , ERROR_LINE()
, ERROR_MESSAGE()
            RETURN
        END
    ELSE IF EXISTS(SELECT hse.Student_id , hse.Exam_id FROM HRStudent s inner join HR_Student_-
Exam hse ON s.Std_id = hse.Student_id AND hse.Student_id = @Student_id INNER JOIN HRExam e ON
e.Exam_id = hse.Exam_id AND hse.Exam_id = @Exam_id )
        BEGIN
            SELECT 'This Student already this exam' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT s.Std_id from HRStudent s inner join HRStudent_Course sc on
s.Std_id = sc.Student_id and sc.Student_id = @Student_id inner join HRExam e ON sc.Course_id =
e.Course_id and e.Exam_id = @Exam_id)
        BEGIN
            SELECT 'This Student not belong to Course of this Exam' , ERROR_LINE() , ERROR_-
MESSAGE()
            RETURN
        END


    ELSE IF EXISTS(SELECT s.Std_name , se.Exam_id from HRStudent s inner join HR_Student_Exam se
on s.Std_id = se.Student_id and se.Student_id = @Student_id inner join HRExam e on se.Exam_id =
e.Exam_id and e.Exam_date = (select Exam_date from HRExam where Exam_id = @Exam_id) and
e.Exam_start < (select Exam_end from HRExam where Exam_id = @Exam_id))
    BEGIN
        SELECT 'This Student has another exam in this date' , ERROR_LINE() , ERROR_MESSAGE()
        RETURN
    END
    ELSE IF NOT EXISTS(select * from HR_Exam_Questions where Exam_Id = @Exam_id)
        BEGIN
            SELECT 'This Exam Not Have Any Questions yet' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
            insert into HR_Student_Exam(Student_ID, Exam_ID)
            values(@Student_ID, @Exam_ID)
            select 'Student Added to Exam Successfully!' as ALERT_MESSAGE
        END

end
GO
```

Uses

[dbo].[HR_Exam_Questions]
[dbo].[HR_Student_Exam]
[dbo].[HRExam]
[dbo].[HRStudent]
[dbo].[HRStudent_Course]

[dbo].[AddExamToStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   proc [dbo].[AddExamToStudent]
            @Student_id int,
            @Exam_id int


as
begin
    IF Not Exists(select Std_id from HRStudent where Std_id = @student_id) or
        Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id)
        BEGIN
            SELECT 'Something is wrong Exam or Student may be not found try again' , ERROR_LINE()
, ERROR_MESSAGE()
            RETURN
        END
    ELSE IF EXISTS(SELECT hse.Student_id , hse.Exam_id FROM HRStudent s inner join HR_Student_-
Exam hse ON s.Std_id = hse.Student_id AND hse.Student_id = @Student_id INNER JOIN HRExam e ON
e.Exam_id = hse.Exam_id AND hse.Exam_id = @Exam_id )
        BEGIN
            SELECT 'This Student already this exam' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
            insert into HR_Student_Exam(Student_ID, Exam_ID)
            values(@Student_ID, @Exam_ID)
            select 'Student Added to Exam Successfully!' as ALERT_MESSAGE
        END


end
GO
```

Uses

[dbo].[HR_Student_Exam]
[dbo].[HRExam]
[dbo].[HRStudent]

[dbo].[AddInstructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Ins_Name	nvarchar(max)	max
@Ins_Age	nvarchar(max)	max
@Ins_Address	nvarchar(max)	max
@Ins_Phone	nvarchar(max)	max

SQL Script

```
create   proc [dbo].[AddInstructor]
      @Ins_Name nvarchar(MAX),
      @Ins_Age nvarchar(MAX),
      @Ins_Address nvarchar(MAX),
      @Ins_Phone nvarchar(MAX)
as
begin
    BEGIN TRY
        INSERT INTO HRInstructoe
        VALUES(@Ins_Name , @Ins_Age , @Ins_Address , @Ins_Phone);
        SELECT 'Instructor is Added Successfully'
    END TRY
    BEGIN CATCH
        SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
    END CATCH
end
GO
```

Uses

[dbo].[HRInstructoe]

[dbo].[AssignCourseToInstructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Class_id	int	4
@Ins_id	int	4
@Course_id	int	4
@Year	int	4

SQL Script

```
create   proc [dbo].[AssignCourseToInstructor]
      @Class_id int,
      @Ins_id int,
      @Course_id int,
      @Year int
as
begin
    IF Not Exists(select Ins_Id from HRInstructoe where Ins_Id = @Ins_id) or
        Not Exists(select Crs_id from HRCousres where Crs_id = @Course_id) or
        Not Exists(select Class_id from HRClass where Class_id = @Class_id)
        BEGIN
            SELECT 'Something is wrong Instructor may be not found try again' , ERROR_LINE() ,
ERROR_MESSAGE()
            RETURN
        END
    ELSE IF Exists(select * from HRClassCourseInstructor where Class_id = @Class_id and Course_Id
= @Course_id and Year = @Year)
            BEGIN
                SELECT 'This Course Already hava instructor in this class in this year' , ERROR_-
LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
    BEGIN
        BEGIN TRY
            INSERT INTO HRClassCourseInstructor
            VALUES(@Class_id , @Course_id , @Year , @Ins_id);
            SELECT 'Instructor is Added Successfully to Course in this Class'
        END TRY
        BEGIN CATCH
            SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
        END CATCH
    END

end
GO
```

Uses

[dbo].[HRClass]
[dbo].[HRClassCourseInstructor]
[dbo].[HRCousres]
[dbo].[HRInstructoe]

[dbo].[AssignStudentToCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Student_id	int	4
@Course_id	int	4

SQL Script

```
create   proc [dbo].[AssignStudentToCourse]
      @Student_id int,
      @Course_id int

as
begin
    IF   Not Exists(select Crs_id from HRCousres where Crs_id = @Course_id) or
        Not Exists(select Std_id from HRStudent where Std_id = @Student_id)
        BEGIN
            SELECT 'Something is wrong Student Or Course may be not found try again' , ERROR_-
LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF Exists(select * from HRStudent_Course where Course_id = @Course_id and Student_id =
@Student_id)
            BEGIN
                SELECT 'This Student Aready join in this course' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
    BEGIN
        BEGIN TRY
            INSERT INTO HRStudent_Course
            VALUES(@Student_id , @Course_id);
            SELECT 'Student is Joined Successfully to Course'
        END TRY
        BEGIN CATCH
            SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
        END CATCH
    END


end
GO
```

Uses

[dbo].[HRCousres]
[dbo].[HRStudent]
[dbo].[HRStudent_Course]

[dbo].[CreateExamProc]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Exam_start	time	5
@Exam_end	time	5
@Exam_date	date	3
@Exam_type	nvarchar(50)	100
@Course_id	int	4
@Instructor_id	int	4
@Intake_id	int	4

SQL Script

```
CREATE   PROCEDURE [dbo].[CreateExamProc]
    @Exam_start TIME,
    @Exam_end TIME,
    @Exam_date DATE,
    @Exam_type NVARCHAR(50),
    @Course_id INT,
    @Instructor_id INT,
    @Intake_id INT
AS
BEGIN
    IF Not Exists(select Crs_id from HRCousres where Crs_id = @Course_id) or
        Not Exists(select Ins_id from HRInstructoe where Ins_Id = @Instructor_id) or
        Not Exists(select Int_id from HRIntake where Int_Id = @Intake_id)
        BEGIN
            SELECT 'Something is wrong Course or Instructor or Intake may be not found try again'
, ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT i.Ins_id from HRInstructoe i inner join HR_Class_Course_Instructor
cci on i.Ins_Id = cci.Instractor_id and cci.Instractor_id = @Instructor_id inner join HRCousres c
ON cci.Course_id = c.Crs_id and cci.Course_id = @Course_id)
    BEGIN
        SELECT 'This Instructor not belong to this Course' , ERROR_LINE() , ERROR_MESSAGE()
        RETURN
    END
    ELSE IF EXISTS(SELECT * from HRExam where Instructor_id = @Instructor_id and Course_id =
@Course_id and Intake_id = @Intake_id and Exam_date = @Exam_date)
    BEGIN
        SELECT 'This Exam is already existed' , ERROR_LINE() , ERROR_MESSAGE()
        RETURN
    END
    ELSE
        BEGIN
                begin try
                    insert into HRExam
                        values(
                                convert(TIME ,  @Exam_start), --convert from varchar to Time
                                convert(TIME ,  @Exam_end), --convert from varchar to Time
                                convert(DATE ,  @Exam_date), --convert from varchar to Date
                                @Exam_type,
                                @Course_id,
                                @Instructor_id,
                                @Intake_id

                                )
                end try
                begin catch
                    select 'Something is wrong Try enter another data' , ERROR_LINE() , ERROR_-
MESSAGE()
                end catch
        END
END

--CreateExamProc '10:00:00.0000000' , '11:00:00.0000000', '2024-01-24', 'Corrective' , 3 , 3 , 1
GO
```

Uses

[dbo].[HR_Class_Course_Instructor]
[dbo].[HRCousres]
[dbo].[HRExam]
[dbo].[HRInstructoe]
[dbo].[HRIntake]

[dbo].[CreateStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@name	nvarchar(50)	100
@age	int	4
@address	nvarchar(50)	100
@phone	int	4
@intakeID	int	4
@classID	int	4

SQL Script

```
CREATE PROC [dbo].[CreateStudent]
    @name NVARCHAR(50),
    @age INT,
    @address NVARCHAR(50),
    @phone INT,
    @intakeID INT,
    @classID INT
AS
BEGIN


    BEGIN TRY
        INSERT INTO [dbo].[Student]
               (Std_name, Std_age, Std_address, Std_phone, Intake_id, class_id)
         VALUES
               (@name, @age, @address, @phone, @intakeID, @classID);

        SELECT 'Student created successfully.';
    END TRY
    BEGIN CATCH
        SELECT 'An error occurred while creating the student. Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO
```

Uses

[dbo].[Student]

[dbo].[DeleteAllQuestionFromExam]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   proc [dbo].[DeleteAllQuestionFromExam]
    @Exam_id int
as
begin
    IF Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id)
        BEGIN
            SELECT 'Something is wrong Exam that you selected may be not found try again' ,
ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(select * from HR_Exam_Questions where Exam_Id = @Exam_id)
        BEGIN
            SELECT 'Exam Not Have Any Questions to update it' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
            BEGIN TRY
                DELETE FROM HR_Exam_Questions
                WHERE Exam_Id = @Exam_id;
                SELECT 'All Answer are Removed'
            END TRY
            BEGIN CATCH
                SELECT 'Something is Wrong' , ERROR_LINE() , ERROR_MESSAGE()
            END CATCH
        END
end
GO
```

Uses

[dbo].[HR_Exam_Questions]
[dbo].[HRExam]

[dbo].[DeleteAssignCourseToInstructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Class_id	int	4
@Course_id	int	4
@Year	int	4

SQL Script

```
create   proc [dbo].[DeleteAssignCourseToInstructor]
      @Class_id int,
      @Course_id int,
      @Year int
as
begin
    IF  Not Exists(select Crs_id from HRCousres where Crs_id = @Course_id) or
        Not Exists(select Class_id from HRClass where Class_id = @Class_id)
        BEGIN
            SELECT 'Something is wrong Instructor may be not found try again' , ERROR_LINE() ,
ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT Exists(select * from HRClassCourseInstructor where Class_id = @Class_id and
Course_Id = @Course_id and Year = @Year)
            BEGIN
                SELECT 'This Course not Have instructor in this class in this year' , ERROR_-
LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
    BEGIN
        BEGIN TRY
            DELETE FROM HRClassCourseInstructor
            where Class_Id = @Class_id and Course_Id = @Course_id and Year = @Year;
            SELECT 'Instructor is Deleted Successfully from Course in this Class'
        END TRY
        BEGIN CATCH
```

```
            SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
        END CATCH
    END

end
GO
```

Uses

[dbo].[HRClass]
[dbo].[HRClassCourseInstructor]
[dbo].[HRCousres]

[dbo].[DeleteAssignStudentToCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Student_id	int	4
@Course_id	int	4

SQL Script

```
create   proc [dbo].[DeleteAssignStudentToCourse]
      @Student_id int,
      @Course_id int

as
begin
    IF  Not Exists(select Crs_id from HRCousres where Crs_id = @Course_id) or
        Not Exists(select Std_id from HRStudent where Std_id = @Student_id)
        BEGIN
            SELECT 'Something is wrong Student Or Course may be not found try again' , ERROR_-
LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT Exists(select * from HRStudent_Course where Course_id = @Course_id and Student_id
= @Student_id)
            BEGIN
                SELECT 'This Student Not join in this course' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
    BEGIN
        BEGIN TRY
            DELETE FROM HRStudent_Course
            WHERE Course_id = @Course_id and Student_id = @Student_id
            SELECT 'Student is Removed Successfully from Course'
        END TRY
        BEGIN CATCH
            SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
        END CATCH
    END


end
GO
```

Uses

[dbo].[HRCousres]
[dbo].[HRStudent]
[dbo].[HRStudent_Course]

[dbo].[DeleteBranch]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   proc [dbo].[DeleteBranch]
        @Branch_id int
as
begin
    IF Not Exists(select Branch_id from HRBranch where Branch_id = @Branch_id)
        BEGIN
            SELECT 'Something is wrong Branch may be not found try again' , ERROR_LINE() ,
ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
            BEGIN TRY
                DELETE FROM HRBranch
                Where Branch_id = @Branch_id;
                    SELECT 'Branch is Deleted Successfully'
                END TRY
                BEGIN CATCH
                    SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
            END CATCH
        END

end

--DeleteBranch 1
GO
```

Uses

[dbo].[HRBranch]

[dbo].[DeleteExamFromStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   proc [dbo].[DeleteExamFromStudent]
            @Student_id int,
            @Exam_id int
as
begin
    IF Not Exists(select Std_id from HRStudent where Std_id = @Student_id) or
        Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id)
        BEGIN
            SELECT 'Something is wrong Exam or Student may be not found try again' , ERROR_LINE()
, ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT hse.Student_id , hse.Exam_id FROM HR_Student_Exam hse where
hse.Student_id = @Student_id and Exam_id = @Exam_id )
        BEGIN
            SELECT 'This Exam not assigned to this Student' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END

    ELSE
        BEGIN
            BEGIN TRY
                DELETE FROM HR_Student_Exam
                WHERE Student_ID = @Student_ID AND Exam_id = @Exam_id;
                SELECT 'Exam Removed From This Student' as ALERT_MESSAGE
            END TRY
            BEGIN CATCH
                SELECT 'Something is wrong try again' as ALERT_MESSAGE
            END CATCH
        END
end
GO
```

Uses

[dbo].[HR_Student_Exam]
[dbo].[HRExam]
[dbo].[HRStudent]

[dbo].[DeleteExamProc]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE   PROCEDURE [dbo].[DeleteExamProc]
    @Exam_id INT
AS
BEGIN
    if Exists(Select Exam_id from Exam where Exam_id = @Exam_id)
        begin
            DELETE FROM HRExam
            WHERE Exam_id = @Exam_id
        end
    else
    begin
        select 'This exam not found must try again by another Exam_id' , ERROR_LINE() , ERROR_-
MESSAGE()
    end

END
GO
```

Uses

[dbo].[Exam]
[dbo].[HRExam]

[dbo].[DeleteInstructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   proc [dbo].[DeleteInstructor]
        @Ins_id int
as
begin
    IF Not Exists(select Ins_Id from HRInstructoe where Ins_Id = @Ins_id)
        BEGIN
            SELECT 'Something is wrong Branch may be not found try again' , ERROR_LINE() ,
ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
            BEGIN TRY
                DELETE FROM HRInstructoe
                Where Ins_Id = @Ins_id;
                    SELECT 'Instructor is Deleted Successfully'
                END TRY
                BEGIN CATCH
                    SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
            END CATCH
        END

end
GO
```

Uses

[dbo].[HRInstructoe]

[dbo].[DeleteStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE PROC [dbo].[DeleteStudent]
    @id INT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY

        IF EXISTS (SELECT 1 FROM Student WHERE Std_ID = @id)
        BEGIN

            DELETE FROM Student
            WHERE Std_ID = @id;

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

GO
```

Uses

[dbo].[Student]

[dbo].[EditStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE PROCEDURE [dbo].[EditStudent]
    @name NVARCHAR(50),
    @email NVARCHAR(50),
    @id INT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY

        IF EXISTS (SELECT 1 FROM Student WHERE Std_ID = @id)
        BEGIN
            UPDATE [dbo].[Student]
            SET Std_name = @name
            WHERE Std_ID = @id;

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

        PRINT 'An error occurred while updating the student information. Error: ' + ERROR_-
MESSAGE();
    END CATCH
```

```
END;


GO
```

Uses

[dbo].[Student]

[dbo].[GetAllExams]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
CREATE   PROCEDURE [dbo].[GetAllExams]
AS
BEGIN
    SELECT *
    FROM HRExam
END
GO
```

Uses

[dbo].[HRExam]

[dbo].[GetDetailsForExam]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--Get Details of Exam by Exam_id
CREATE   PROCEDURE [dbo].[GetDetailsForExam] @Exam_id INT
AS
BEGIN
    if exists(SELECT Exam_id from Exam where Exam_id = @Exam_id)
    begin
        SELECT *
        FROM Exam
        WHERE Exam_id = @Exam_Id
    end
    else
        begin
            select 'This exam not found must try again by another Exam_id' , ERROR_LINE() ,
ERROR_MESSAGE()
        end
END

-- GetDetailsForExam 1
GO
```

Uses

[dbo].[Exam]

[dbo].[GetExamResultForStudent]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE   PROCEDURE [dbo].[GetExamResultForStudent]
    @Student_id INT,
    @Exam_id INT
AS
BEGIN

    IF Not Exists(select Std_id from HRStudent where Std_id = @student_id) or
        Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id)
        BEGIN
            SELECT 'Something is wrong Exam or Student may be not found try again' , ERROR_LINE()
, ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT hse.Student_id , hse.Exam_id FROM HRStudent s inner join HR_-
Student_Exam hse ON s.Std_id = hse.Student_id AND hse.Student_id = @Student_id INNER JOIN HRExam
e ON e.Exam_id = hse.Exam_id AND hse.Exam_id = @Exam_id )
        BEGIN
            SELECT 'This Student is not have this exam' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT Student_id FROM HR_Exam_Questions_Student WHERE Student_id =
@Student_id AND Exam_id = @Exam_id)
    BEGIN
        SELECT 'This Student not answer this exam yet' , ERROR_LINE() , ERROR_MESSAGE()
        RETURN
    END
    ELSE
        BEGIN
            -- Create a temporary table to hold student exam results
            CREATE TABLE #ExamAnswersResults_TMP(
                [Exam_id] [int] NOT NULL,
                [Question_id] [int] NOT NULL,
                [Student_id] [int] NOT NULL,
                [Student_Answer] [nvarchar](50) NOT NULL,
            )

            INSERT INTO #ExamAnswersResults_TMP
            SELECT * FROM Exam_Questions_Student WHERE Student_id =  @Student_id AND Exam_id = 
@Exam_id

            DECLARE @Question_id INT
            DECLARE @Total_result INT = 0;
            DECLARE @Student_Answer NVARCHAR(50)
            DECLARE @Correct_Answer NVARCHAR(50)

            DECLARE CalculateAnswerExamCursor CURSOR FOR
            SELECT tr.Question_id, tr.Student_Answer , q.Correct_answer
            FROM #ExamAnswersResults_TMP tr
            INNER JOIN Question q ON tr.Question_id = q.Question_id

            OPEN CalculateAnswerExamCursor
            FETCH NEXT FROM CalculateAnswerExamCursor INTO @Question_id, @Student_Answer ,
@Correct_Answer

            --Get total degree for course
            DECLARE @Exam_Total_degree int = 0;

            WHILE @@FETCH_STATUS = 0
            BEGIN

                DECLARE @Question_Degree INT --Get question degree
                SELECT @Question_Degree = Degree --set degree in varriable
                FROM HR_Exam_Questions
                WHERE Exam_ID = @Exam_id AND Question_id = @Question_id

                SET @Exam_Total_degree = @Exam_Total_degree +  @Question_Degree;

                IF @Student_Answer = @Correct_Answer
                    BEGIN
                        SET @Total_result = @Total_result +  @Question_Degree
                    END
                ELSE
                    BEGIN
                        SET @Total_result = @Total_result + 0 -- Set the question grade if none
of the conditions match
                    END
                FETCH NEXT FROM CalculateAnswerExamCursor INTO @Question_id, @Student_Answer ,
@Correct_Answer
            END

             -- Update the question result
                UPDATE HR_Student_Exam SET Exam_result = @Total_result
                WHERE Student_id = @student_id AND Exam_id = @Exam_id

                DECLARE @Std_name NVARCHAR(50);
                DECLARE @Crs_name NVARCHAR(50);

                SELECT @Std_name = Std_name FROM HRStudent WHERE Std_id = @Student_id;
                SELECT @Crs_name = c.Crs_name FROM HRCousres c INNER JOIN HRExam e
                ON c.crs_id = e.Course_id AND e.Exam_id = @Exam_id;

                SELECT Concat('Result for Student : ' , @Std_name , ' of ' , @Crs_name , ' Course
is: ' , @Total_result , '/' ,  @Exam_Total_degree);
            CLOSE CalculateAnswerExamCursor
            DEALLOCATE CalculateAnswerExamCursor
            -- Drop temporary table after complete calculation
            DROP TABLE #ExamAnswersResults_TMP
        END
END
GO
```

Uses

[dbo].[Exam_Questions_Student]
[dbo].[Question]
[dbo].[HR_Exam_Questions]
[dbo].[HR_Exam_Questions_Student]
[dbo].[HR_Student_Exam]
[dbo].[HRCousres]
[dbo].[HRExam]
[dbo].[HRStudent]

[dbo].[GetExamsByCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE   PROCEDURE [dbo].[GetExamsByCourse]
    @Course_id INT
AS
BEGIN
    SELECT *
    FROM Exam
    WHERE Course_id = @Course_id
END
GO
```

Uses

[dbo].[Exam]

[dbo].[GetExamsByDate]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Exam_date	date	3

SQL Script

```
--Get Exams by date
CREATE   PROCEDURE [dbo].[GetExamsByDate]
    @Exam_date Date
AS
BEGIN
    SELECT *
    FROM Exam
    WHERE DAY(Exam_date) = DAY(@Exam_date)
END
GO
```

Uses

[dbo].[Exam]

[dbo].[GetExamsByDateRange]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Exam_start	time	5
@Exam_end	time	5

SQL Script

```
CREATE   PROCEDURE [dbo].[GetExamsByDateRange]
    @Exam_start TIME,
    @Exam_end TIME
AS
BEGIN
    SELECT *
    FROM Exam
    WHERE Exam_start BETWEEN @Exam_start AND @Exam_end
END
GO
```

Uses

[dbo].[Exam]

[dbo].[GetTotalDegreeForStudentExam]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE   PROCEDURE [dbo].[GetTotalDegreeForStudentExam]
    @Student_id INT,
    @Exam_id INT
AS
BEGIN

    IF Not Exists(select Std_id from HRStudent where Std_id = @student_id) or
        Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id)
        BEGIN
            SELECT 'Something is wrong Exam or Student may be not found try again' , ERROR_LINE()
, ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT hse.Student_id , hse.Exam_id FROM HRStudent s inner join HR_-
Student_Exam hse ON s.Std_id = hse.Student_id AND hse.Student_id = @Student_id INNER JOIN HRExam
e ON e.Exam_id = hse.Exam_id AND hse.Exam_id = @Exam_id )
        BEGIN
            SELECT 'This Student is not have this exam' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT Student_id FROM HR_Exam_Questions_Student WHERE Student_id =
@Student_id AND Exam_id = @Exam_id)
        BEGIN
            SELECT 'This Student not answer this exam yet' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
             DECLARE @Exam_result INT;
             SELECT @Exam_result = Exam_result FROM HR_Student_Exam WHERE Student_id =
@Student_id and Exam_id = @Exam_id;

             DECLARE @Course_MinDegree INT ,  @Crs_name NVARCHAR(50);
             SELECT @Course_MinDegree = Crs_minDegree , @Crs_name = Crs_name FROM Course c inner
join Exam e ON c.Crs_id = e.Course_id WHERE e.Exam_id = @Exam_id;

             DECLARE @Std_name NVARCHAR(50);
             SELECT @Std_name = Std_name FROM HRStudent WHERE Std_id = @Student_id;




            IF @Exam_result >= @Course_MinDegree
                BEGIN
                    select CONCAT('Result for Student : ' , @Std_name , ' of ' , @Crs_name , '
Course is: ' , @Exam_result , ' Passed')
                END
            ELSE
                BEGIN
                    select CONCAT('Result for Student : ' , @Std_name , ' of ' , @Crs_name , '
Course is: ' , @Exam_result , ' Failed')
                END
        END


END
GO
```

Uses

[dbo].[Course]
[dbo].[Exam]
[dbo].[HR_Exam_Questions_Student]
[dbo].[HR_Student_Exam]
[dbo].[HRExam]
[dbo].[HRStudent]

[dbo].[GetTotalDegreeForStudentExamAndStatus]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--Get total degree For student and status
CREATE   PROCEDURE [dbo].[GetTotalDegreeForStudentExamAndStatus]
    @Student_id INT,
    @Exam_id INT
AS
BEGIN

    IF Not Exists(select Std_id from HRStudent where Std_id = @student_id) or
        Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id)
        BEGIN
            SELECT 'Something is wrong Exam or Student may be not found try again' , ERROR_LINE()
, ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT hse.Student_id , hse.Exam_id FROM HRStudent s inner join HR_-
Student_Exam hse ON s.Std_id = hse.Student_id AND hse.Student_id = @Student_id INNER JOIN HRExam
e ON e.Exam_id = hse.Exam_id AND hse.Exam_id = @Exam_id )
        BEGIN
            SELECT 'This Student is not have this exam' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT Student_id FROM HR_Exam_Questions_Student WHERE Student_id =
@Student_id AND Exam_id = @Exam_id)
        BEGIN
            SELECT 'This Student not answer this exam yet' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
```

```
             DECLARE @Exam_result INT;
             SELECT @Exam_result = Exam_result FROM HR_Student_Exam WHERE Student_id =
@Student_id and Exam_id = @Exam_id;

             DECLARE @Course_MinDegree INT ,  @Crs_name NVARCHAR(50);
             SELECT @Course_MinDegree = Crs_minDegree , @Crs_name = Crs_name FROM Course c inner
join Exam e ON c.Crs_id = e.Course_id WHERE e.Exam_id = @Exam_id;

             DECLARE @Std_name NVARCHAR(50);
             SELECT @Std_name = Std_name FROM HRStudent WHERE Std_id = @Student_id;




            IF @Exam_result >= @Course_MinDegree
                BEGIN
                    select CONCAT('Result for Student : ' , @Std_name , ' of ' , @Crs_name , '
Course is: ' , @Exam_result , ' Passed')
                END
            ELSE
                BEGIN
                    select CONCAT('Result for Student : ' , @Std_name , ' of ' , @Crs_name , '
Course is: ' , @Exam_result , ' Failed')
                END
        END


END

--GetTotalDegreeForStudentExamAndStatus 30 , 1
GO
```

Uses

[dbo].[Course]
[dbo].[Exam]
[dbo].[HR_Exam_Questions_Student]
[dbo].[HR_Student_Exam]
[dbo].[HRExam]
[dbo].[HRStudent]

[dbo].[SelectQuestionsManualForExam]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   proc [dbo].[SelectQuestionsManualForExam]
    @Exam_id int,
    @Question_Selected_Id int,
    @Degree int
as
begin
    IF Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id) or
        Not Exists(select Question_id from HRQuestions where Question_id = @Question_Selected_Id)
        BEGIN
            SELECT 'Something is wrong Exam or Question that you selected may be not found try
again' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END

    declare @course_id int;
    declare @crs_maxDegree int;

    select @course_id = Course_id from HRExam where Exam_id = @Exam_id;
    select @crs_maxDegree = Crs_maxDegree from HRCousres where Crs_id = @course_id;

    declare @questions_course table(Question_id int);

    insert into @questions_course
    select Question_id from  dbo.getQuestionsForSpecificCourse(@course_id)

    IF NOT EXISTS (select * from @questions_course where Question_id = @Question_Selected_Id)
        BEGIN
```

```
            SELECT 'This Question Not Belong to Course of Exam' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
        --check if Degree of Questions More than Degree of Exam
        Declare @totalDegree int;
        select @totalDegree = sum(Degree) from HR_Exam_Questions where Exam_Id = @Exam_id
        set @totalDegree = @totalDegree + @Degree;
            IF @totalDegree > @crs_maxDegree
                BEGIN
                    select CONCAT('Sum of Questions Degree more than Exam Degree ' , @totalDegree
, ' Max is ' , @crs_maxDegree )
                    return
                END
            Else
                BEGIN
                    BEGIN TRY
                        Insert into HR_Exam_Questions
                        values(@Exam_id , @Question_Selected_Id , @Degree);
                        select 'Question Added to Exam'
                    END TRY
                    BEGIN CATCH
                        select 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE();
                    END CATCH

                END
        END
end
GO
```

Uses

[dbo].[getQuestionsForSpecificCourse]
[dbo].[HR_Exam_Questions]
[dbo].[HRCousres]
[dbo].[HRExam]
[dbo].[HRQuestions]

[dbo].[SelectQuestionsRandomForExam]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
create   proc [dbo].[SelectQuestionsRandomForExam]
    @Exam_id int,
    @Number_Of_Question int
as
begin
    IF Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id)
        BEGIN
            SELECT 'Something is wrong Exam that you selected may be not found try again' ,
ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END

    declare @course_id int;
    declare @crs_maxDegree int;

    select @course_id = Course_id from HRExam where Exam_id = @Exam_id;
    select @crs_maxDegree = Crs_maxDegree from HRCousres where Crs_id = @course_id;

    declare @questions_course table(Question_id int);
    insert into @questions_course
    select top(@Number_Of_Question) Question_id from  dbo.getQuestionsForSpecific-
Course(@course_id) order by NEWID()

    Declare @Number_Of_Questions_After_Randomizing int;
    Declare @Degree_Each_Question int;

    select @Number_Of_Questions_After_Randomizing = Count(Question_id) from @questions_course;
    set @Degree_Each_Question = @crs_maxDegree / @Number_Of_Questions_After_Randomizing;
    Declare @Question_id_row int;

    Declare RandomCursor CURSOR
    for select Question_id from @questions_course;

    OPEN RandomCursor;
    FETCH RandomCursor INTO @Question_id_row;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        BEGIN TRY
            Insert into HR_Exam_Questions
            values(@Exam_id , @Question_id_row , @Degree_Each_Question);
            select 'Question Added to Exam'
            FETCH RandomCursor INTO @Question_id_row;
        END TRY
        BEGIN CATCH
            select 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE();
        END CATCH
    END
end
GO
```

Uses

[dbo].[getQuestionsForSpecificCourse]
[dbo].[HR_Exam_Questions]
[dbo].[HRCousres]
[dbo].[HRExam]

[dbo].[UpdateAssignCourseToInstructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Class_id	int	4
@Ins_id	int	4
@Course_id	int	4
@Year	int	4
@Class_New_id	int	4
@Ins_New_id	int	4
@Course_New_id	int	4
@Year_New	int	4

SQL Script

```
create   proc [dbo].[UpdateAssignCourseToInstructor]
      @Class_id int,
      @Ins_id int,
      @Course_id int,
      @Year int,
      @Class_New_id int,
      @Ins_New_id int,
      @Course_New_id int,
      @Year_New int
as
begin
    IF Not Exists(select Ins_Id from HRInstructoe where Ins_Id = @Ins_New_id) or
        Not Exists(select Crs_id from HRCousres where Crs_id = @Course_New_id) or
        Not Exists(select Class_id from HRClass where Class_id = @Class_New_id)
        BEGIN
            SELECT 'Something is wrong Instructor may be not found try again' , ERROR_LINE() ,
ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT Exists(select * from HRClassCourseInstructor where Class_id = @Class_id and
Course_Id = @Course_id and Year = @Year)
```

```
            BEGIN
                SELECT 'This Instructor Not Assigned to this Course' , ERROR_LINE() , ERROR_-
MESSAGE()
            RETURN
        END
    ELSE
    BEGIN
        BEGIN TRY
            UPDATE HRClassCourseInstructor
            set Class_Id = @Class_New_id , Course_Id = @Course_New_id , Year = @Year_New ,
Instractor_ID =  @Ins_New_id
            where Class_Id = @Class_id and Instractor_ID = @Ins_id and Course_Id = @Course_id and
Year = @Year;
            SELECT 'Instructor is Updated Successfully to Course in this Class'
        END TRY
        BEGIN CATCH
            SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
        END CATCH
    END

end
GO
```

Uses

[dbo].[HRClass]
[dbo].[HRClassCourseInstructor]
[dbo].[HRCousres]
[dbo].[HRInstructoe]

[dbo].[UpdateBranch]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Branch_id	int	4
@Branch_Name	nvarchar(max)	max
@Branch_Address	nvarchar(max)	max
@Branch_Phone	nvarchar(max)	max

SQL Script

```
create   proc [dbo].[UpdateBranch]
        @Branch_id int,
        @Branch_Name nvarchar(MAX),
        @Branch_Address nvarchar(MAX),
        @Branch_Phone nvarchar(MAX)
as
begin
    IF Not Exists(select Branch_id from HRBranch where Branch_id = @Branch_id)
        BEGIN
            SELECT 'Something is wrong Branch may be not found try again' , ERROR_LINE() ,
ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
            BEGIN TRY
                UPDATE HRBranch
                SET Branch_name = @Branch_Name , Branch_address = @Branch_Address , Branch_phone
= @Branch_Phone
                Where Branch_id = @Branch_id;
                    SELECT 'Branch is Updated Successfully'
                END TRY
                BEGIN CATCH
                    SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
            END CATCH
        END
```

```
end

--UpdateBranch 3 , 'Alex' , 'Alex 3agamy' , '01225693656'
GO
```

Uses

[dbo].[HRBranch]

[dbo].[UpdateExamProc]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Exam_Id	int	4
@Exam_start	time	5
@Exam_end	time	5
@Exam_date	date	3
@Exam_type	nvarchar(50)	100
@Course_id	int	4
@Instructor_id	int	4
@Intake_id	int	4

SQL Script

```
--Exam Update by Exam_id
CREATE   PROCEDURE [dbo].[UpdateExamProc]
    @Exam_Id INT,
    @Exam_start TIME,
    @Exam_end TIME,
    @Exam_date DATE,
    @Exam_type NVARCHAR(50),
    @Course_id INT,
    @Instructor_id INT,
    @Intake_id INT
AS
BEGIN
    IF Not Exists(select Crs_id from HRCousres where Crs_id = @Course_id) or
        Not Exists(select Ins_id from HRInstructoe where Ins_Id = @Instructor_id) or
        Not Exists(select Int_id from HRIntake where Int_Id = @Intake_id)
        BEGIN
            SELECT 'Something is wrong Course or Instructor or Intake may be not found try again'
, ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(SELECT i.Ins_id from HRInstructoe i inner join HR_Class_Course_Instructor
cci on i.Ins_Id = cci.Instractor_id and cci.Instractor_id = @Instructor_id inner join HRCousres c
ON cci.Course_id = c.Crs_id and cci.Course_id = @Course_id)
    BEGIN
        SELECT 'This Instructor not belong to this Course' , ERROR_LINE() , ERROR_MESSAGE()
        RETURN
    END
    ELSE IF Not EXISTS(SELECT * from HRExam where Instructor_id = @Instructor_id and Course_id =
@Course_id and Intake_id = @Intake_id and Exam_date = @Exam_date)
    BEGIN
        SELECT 'This Exam is not existed' , ERROR_LINE() , ERROR_MESSAGE()
        RETURN
    END
    ELSE
    BEGIN
            begin try
                --check if exam already exists to do your operations
                if Exists(select Exam_id from HRExam where Exam_id = @Exam_Id)
                    begin
                        UPDATE HRExam
                        SET Exam_start =  convert(TIME ,  @Exam_start), --convert from varchar to
Time
                            Exam_end = convert(TIME ,  @Exam_end), --convert from varchar to Time
                            Exam_date = convert(DATE ,  @Exam_date), --convert from varchar to
Date
                            Exam_type = @Exam_type,
                            Course_id = @Course_id,
                            Instructor_id = @Instructor_id,
                            Intake_id = @Intake_id
                        WHERE Exam_id = @Exam_Id
                    end
                else
                    begin
                        select 'This exam not found must try again by another Exam_id' , ERROR_-
LINE() , ERROR_MESSAGE()
                    end
            end try
            begin catch
                select 'Something is wrong Try enter another data' , ERROR_LINE() , ERROR_-
MESSAGE()
            end catch
    END

END

--UpdateExamProc 1 , '08:00:00.0000000' , '09:00:00.0000000', '2024-01-17', 'Normal' , 3 , 2 , 1

GO
```

Uses

[dbo].[HR_Class_Course_Instructor]
[dbo].[HRCousres]
[dbo].[HRExam]

[dbo].[HRInstructoe]

[dbo].[HRIntake]

[dbo].[UpdateInstructor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Ins_id	int	4
@Ins_Name	nvarchar(max)	max
@Ins_Age	nvarchar(max)	max
@Ins_Address	nvarchar(max)	max
@Ins_Phone	nvarchar(max)	max

SQL Script

```
create   proc [dbo].[UpdateInstructor]
      @Ins_id int,
      @Ins_Name nvarchar(MAX),
      @Ins_Age nvarchar(MAX),
      @Ins_Address nvarchar(MAX),
      @Ins_Phone nvarchar(MAX)
as
begin
    IF Not Exists(select Ins_Id from HRInstructoe where Ins_Id = @Ins_id)
        BEGIN
            SELECT 'Something is wrong Instructor may be not found try again' , ERROR_LINE() ,
ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
            BEGIN TRY
                UPDATE HRInstructoe
                SET Ins_name = @Ins_Name , Ins_Age = @Ins_Age , Ins_Address = @Ins_Address ,
Ins_Phone = @Ins_Phone
                Where Ins_Id = @Ins_id;
                    SELECT 'Instructor is Updated Successfully'
                END TRY
                BEGIN CATCH
                    SELECT 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE()
            END CATCH
        END

end

--UpdateBranch 3 , 'Alex' , 'Alex 3agamy' , '01225693656'
GO
```

Uses

[dbo].[HRInstructoe]

[dbo].[UpdateQuestionsForExam]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--update Exam Questions
create   proc [dbo].[UpdateQuestionsForExam]
    @Exam_id int,
    @Question_Old_id int,
    @Question_New_id int,
    @Degree int
as
begin
    IF Not Exists(select Exam_id from HRExam where Exam_id = @Exam_id) or
        Not Exists(select @Question_Old_id from HRQuestions where Question_id = @Question_Old_id)
or
        Not Exists(select @Question_New_id from HRQuestions where Question_id = @Question_New_id)
        BEGIN
            SELECT 'Something is wrong Exam or Questions that you selected may be not found try
again' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE IF NOT EXISTS(select * from HR_Exam_Questions where Exam_Id = @Exam_id and Question_id =
@Question_Old_id)
        BEGIN
            SELECT 'Exam Not Have This Question to update it' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    declare @course_id int;
    declare @crs_maxDegree int;

    select @course_id = Course_id from HRExam where Exam_id = @Exam_id;
    select @crs_maxDegree = Crs_maxDegree from HRCousres where Crs_id = @course_id;

    declare @questions_course table(Question_id int);

    insert into @questions_course
    select Question_id from  dbo.getQuestionsForSpecificCourse(@course_id)

    IF NOT EXISTS (select * from @questions_course where Question_id = @Question_New_id)
        BEGIN
            SELECT 'This Question Not Belong to Course of Exam' , ERROR_LINE() , ERROR_MESSAGE()
            RETURN
        END
    ELSE
        BEGIN
        --check if Degree of Questions More than Degree of Exam
        Declare @totalDegree int;
        select @totalDegree = sum(Degree) from HR_Exam_Questions where Exam_Id = @Exam_id

        Declare @UpdatedDegree int;
        select @UpdatedDegree = Degree from HR_Exam_Questions where Exam_Id = @Exam_id and
Question_Id = @Question_New_id;
        set @totalDegree = @totalDegree - @UpdatedDegree + @Degree;
            IF @totalDegree > @crs_maxDegree
                BEGIN
                    select CONCAT('Sum of Questions Degree more than Exam Degree ' , @totalDegree
, ' Max is ' , @crs_maxDegree )
                    return
                END
            Else
                BEGIN
                    BEGIN TRY
                        Update HR_Exam_Questions
                        set Question_Id = @Question_New_id , Degree = @Degree
                        where Question_Id = @Question_Old_id and Exam_Id = @Exam_id;
                        select 'Question Updated to Exam'
                    END TRY
                    BEGIN CATCH
                        select 'Something is wrong' , ERROR_LINE() , ERROR_MESSAGE();
                    END CATCH

                END
        END
end
GO
```

Uses

[dbo].[getQuestionsForSpecificCourse]
[dbo].[HR_Exam_Questions]
[dbo].[HRCousres]
[dbo].[HRExam]

[dbo].[HRQuestions]

Table-valued Functions

Objects

Name
dbo.getQuestionsForSpecificCourse

[dbo].[getQuestionsForSpecificCourse]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
--Function get Questions for specific course by course_id
create   function [dbo].[getQuestionsForSpecificCourse](@Course_id int)
returns table
as
return
(
    select *
    from Question
    where Course_id = @Course_id
)
GO
```

Uses

[dbo].[Question]

Used By

[dbo].[SelectQuestionsManualForExam]
[dbo].[SelectQuestionsRandomForExam]
[dbo].[UpdateQuestionsForExam]

Scalar-valued Functions

Objects

Name
dbo.GetStudentTotalDegreeFromHisAnswers

[dbo].[GetStudentTotalDegreeFromHisAnswers]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

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

SQL Script

```
CREATE FUNCTION [dbo].[GetStudentTotalDegreeFromHisAnswers] (@student_id int , @Exam_id int)
returns NVARCHAR(MAX)
begin
        if Not Exists(select Std_id from Student where Std_id = @student_id) or
        Not Exists(select Exam_id from Exam where Exam_id = @Exam_id)
        begin
            declare @message NVARCHAR(MAX);
            SET @message = 'Something is wrong Exam or Student may be not found try again';
            return @message;
        end

        return 'no any proplem'

end
GO
```

Uses

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

Database Triggers

Objects

Name
ddlt_PreventDDLDropTable
PreventDropTable_TRIGGER

ddlt_PreventDDLDropTable

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
CREATE   Trigger [ddlt_PreventDDLDropTable]
ON DATABASE FOR DROP_TABLE
AS

SELECT
 EventType = EVENTDATA().value('(EVENT_INSTANCE/EventType)[1]', 'sysname'),
 PostTime = EVENTDATA().value('(EVENT_INSTANCE/PostTime)[1]', 'datetime'),
 LoginName = EVENTDATA().value('(EVENT_INSTANCE/LoginName)[1]', 'sysname'),
 UserName = EVENTDATA().value('(EVENT_INSTANCE/UserName)[1]', 'sysname'),
 DatabaseName = EVENTDATA().value('(EVENT_INSTANCE/DatabaseName)[1]', 'sysname'),
 SchemaName = EVENTDATA().value('(EVENT_INSTANCE/SchemaName)[1]', 'sysname'),
 ObjectName = EVENTDATA().value('(EVENT_INSTANCE/ObjectName)[1]', 'sysname'),
 ObjectType = EVENTDATA().value('(EVENT_INSTANCE/ObjectType)[1]', 'sysname'),
 CommandText = EVENTDATA().value('(EVENT_INSTANCE//TSQLCommand[1]/CommandText)[1]',
'nvarchar(max)')

RAISERROR ('You can not drop table in this database', 10, 1);
ROLLBACK;
GO
```

PreventDropTable_TRIGGER

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
--Trigger To prevent drop any table from system

CREATE   Trigger [PreventDropTable_TRIGGER]
ON DATABASE FOR DROP_TABLE
AS

SELECT
 EventType = EVENTDATA().value('(EVENT_INSTANCE/EventType)[1]', 'sysname'),
 PostTime = EVENTDATA().value('(EVENT_INSTANCE/PostTime)[1]', 'datetime'),
 LoginName = EVENTDATA().value('(EVENT_INSTANCE/LoginName)[1]', 'sysname'),
 UserName = EVENTDATA().value('(EVENT_INSTANCE/UserName)[1]', 'sysname'),
 DatabaseName = EVENTDATA().value('(EVENT_INSTANCE/DatabaseName)[1]', 'sysname'),
 SchemaName = EVENTDATA().value('(EVENT_INSTANCE/SchemaName)[1]', 'sysname'),
 ObjectName = EVENTDATA().value('(EVENT_INSTANCE/ObjectName)[1]', 'sysname'),
 ObjectType = EVENTDATA().value('(EVENT_INSTANCE/ObjectType)[1]', 'sysname'),
 CommandText = EVENTDATA().value('(EVENT_INSTANCE//TSQLCommand[1]/CommandText)[1]',
'nvarchar(max)')

RAISERROR ('You can not drop table in this database', 10, 1);
ROLLBACK;

--DROP TABLE [dbo].[Branch]

GO
```

Users

Objects

Name
Admin
instructor
Manager
Student
TrainingManager

Admin

Properties

Property	Value
Type	SqlUser
Login Name	Admin
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

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

instructor

Properties

Property	Value
Type	SqlUser
Login Name	instructor
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

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

Used By

instructor_role

Manager

Properties

Property	Value
Type	SqlUser
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
CREATE USER [Manager] WITHOUT LOGIN
GO
```

Student

Properties

Property	Value
Type	SqlUser
Login Name	Student
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

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

Used By

student_role

TrainingManager

Properties

Property	Value
Type	SqlUser
Login Name	TrainingManager
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

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

Used By

training_manager_role

Database Roles

Objects

Name
admin_role
instructor_role
student_role
training_manager_role

admin_role

Properties

Property	Value
Owner	dbo

Database Level Permissions

Type	Action
CREATE PROCEDURE	Grant
CREATE TABLE	Grant

SQL Script

```
CREATE ROLE [admin_role]
AUTHORIZATION [dbo]
GO
GRANT ALTER ANY USER TO [admin_role]
GRANT CREATE PROCEDURE TO [admin_role]
GRANT CREATE TABLE TO [admin_role]
```

instructor_role

Properties

Property	Value
Owner	dbo

Members

- instructor

SQL Script

```
CREATE ROLE [instructor_role]
AUTHORIZATION [dbo]
GO
ALTER ROLE [instructor_role] ADD MEMBER [instructor]
GO
```

Uses

instructor

student_role

Properties

Property	Value
Owner	dbo

Members

- Student

SQL Script

```
CREATE ROLE [student_role]
AUTHORIZATION [dbo]
GO
ALTER ROLE [student_role] ADD MEMBER [Student]
GO
```

Uses

Student

training_manager_role

Properties

Property	Value
Owner	dbo

Members

- TrainingManager

Database Level Permissions

Type	Action
CREATE PROCEDURE	Grant
CREATE TABLE	Grant

SQL Script

```
CREATE ROLE [training_manager_role]
AUTHORIZATION [dbo]
GO
ALTER ROLE [training_manager_role] ADD MEMBER [TrainingManager]
GO
GRANT CREATE PROCEDURE TO [training_manager_role]
GRANT CREATE TABLE TO [training_manager_role]
```

Uses

TrainingManager