ITI Examination System

Team 2

1. **Hany Abdo Saad**
2. **Mostafa Bolbol Ramadan**
3. **Omar Mohamed Araby**
4. **Yasmine Mohamed Ibrahim**
5. **Ahmed Sobhi Abdelhamid**

Documentation

Examination System Description

System Overview

The examination system is designed to automate online exams and manage related data efficiently using a relational database. The system supports exam creation, student management, grading, and reporting for ITI staff.

Core Features

- Exam creation and question management.
- Exam answers submission by students.
- Automated exam correction and grading.
- Student enrollment in courses and exams.
- Instructor management and course assignment.

Database Operations

- Perform CRUD (Create, Read, Update, Delete) operations on all tables.
- Use stored procedures for optimized data retrieval and manipulation.

Reporting

Generate reports for:

- Student details filtered by department.
- Student grades across courses.
- Instructor's courses and enrolled students per course.
- Course details, including topics.
- Exam details (questions and choices).
- Student-specific exam answers and grades.

Student Management

- **Student Table:** Stores student personal and academic data.
- **Std_Course Table:** Links students to their enrolled courses.
- **Std_Exam Table:** Records students' grades and exam details.
- **Std_ExamAnswer Table:** Tracks students' answers for each question.

Instructor Management

- **Instructor Table:** Maintains details of instructors, including salary and hire date.
- **Ins_Course Table:** Maps instructors to the courses they teach.

Course and Topics

- **Course Table:** Contains information on courses and their durations.
- **Topic Table:** Lists topics for each course.

Exams and Questions

- **Exam Table:** Defines exams, their duration, and model type.
- **Ques_Exam Table:** Associates exams with their respective questions.
- **Question Table:** Holds details of exam questions, their type, and answers.
- **Question_Type Table:** Includes optional answers for multiple-choice questions.

Department Management

Department Table: Organizes departments with managers and contact details.

Data Flow

1. **Exam Lifecycle**
   - **Creation:** Admins or instructors add courses, topics, and questions to generate exams.
   - **Enrollment:** Students enroll in courses and exams.
   - **Answer Submission:** Students submit answers during exams, stored in the Std_ExamAnswer table.
   - **Correction:** Automated grading evaluates the answers.
   - **Results:** Grades are saved in the Std_Exam table.

2. **Reporting Process**
   - Reports utilize stored procedures to fetch data, ensuring efficiency and security.

Requirements

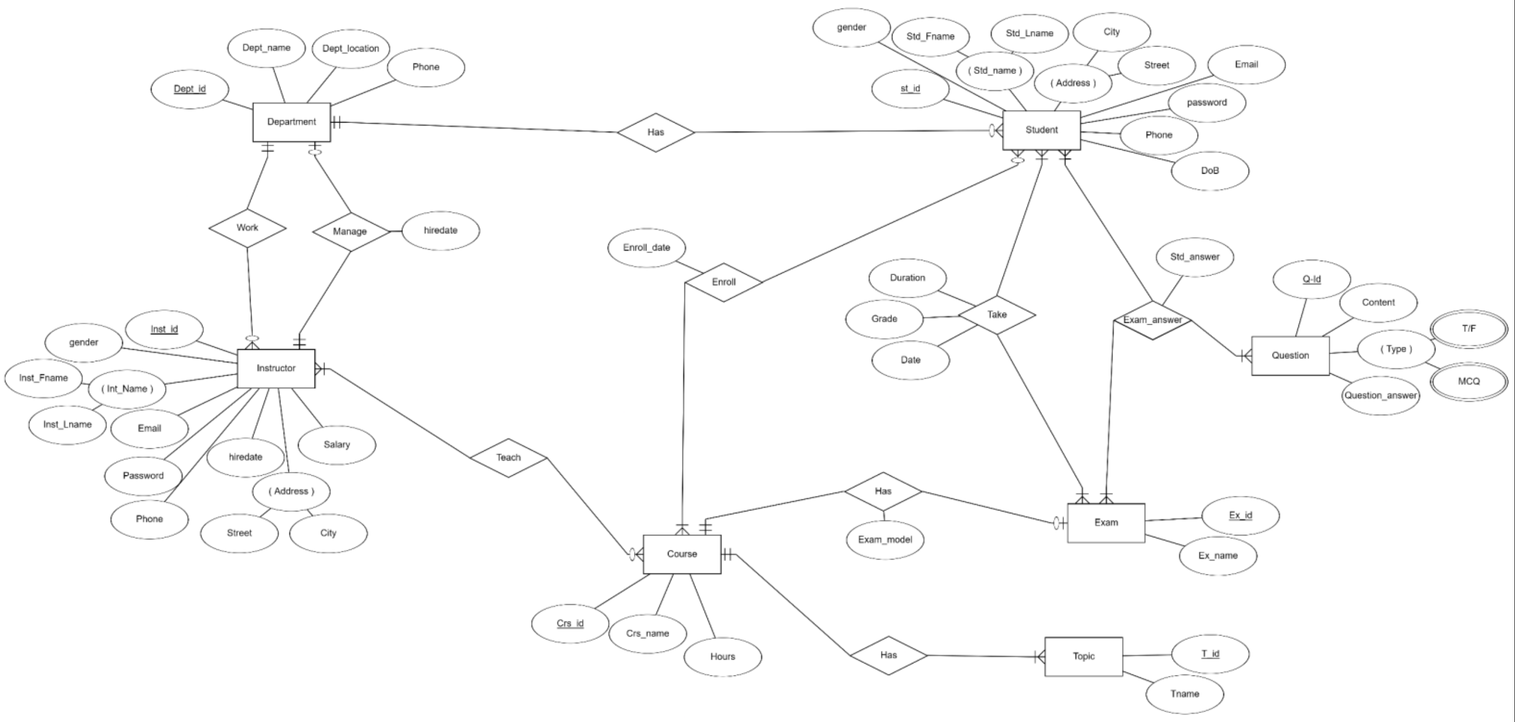
Relationships

1. **Department-Manager Relationship**
    - Each department is managed by one instructor (1:1 relationship).
    - The manager's hire date is stored as part of the department attributes.
2. **Department-Student Relationship**
    - Each department can have many students, but each student belongs to one department (1:N relationship).
3. **Department-Instructor Relationship**
    - Each department can have many instructors, but each instructor belongs to one department (1:N relationship).
4. **Course-Department Relationship**
    - Each course is offered by one department, but a department can offer multiple courses (1:N relationship).
5. **Course-Topic Relationship**

- Each course can have multiple topics, but each topic belongs to one course (1:N relationship).

6. **Exam-Course Relationship**
   - Each exam is associated with one course, but the course may be has one Exam(1:1 relationship).
7. **Question-Exam Relationship**
   - Each exam consists of multiple questions, and each question belongs to multiple exam (M:N relationship).
8. **Student-Course Relationship**
   - Students can enroll in multiple courses, and each course can have multiple students (M:N relationship).
9. **Student-Exam Relationship**
   - Students can take multiple exams, and each exam can be taken by multiple students (M:N relationship).
10. **Student-Question and Exam Relationship**
    - Students can answer multiple questions in an exam, and each question can be answered by multiple students (ternary relationship).

➢ ERD


gender
Std_Fname
Std_Lname
City
(Std_name)
Street
Email
(Address)
st_id
password
Student
Phone
DoB
Dept_name
Dept_location
Phone
Dept_id
Department
Has
Work
Manage
hiredate
Enroll_date
Enroll
Std_answer
Duration
Q-Id
Grade
Take
Content
Exam_answer
T/F
Date
Question
(Type)
MCQ
Question_answer
gender
Inst_id
Inst_Fname
(Int_Name)
Instructor
Inst_Lname
Email
Salary
hiredate
Password
(Address)
Phone
Street
City
Teach
Has
Exam_model
Exam
Ex_id
Ex_name
Course
Crs_id
Crs_name
Hours
Has
Topic
T_id
Tname

➢ Entities and Attributes

1. **Student**
   - Attributes:
     - **std_id (Primary Key)**: A unique identifier for each student.
     - **std_fname**: First name of the student.
     - **std_lname**: Last name of the student.
     - **std_DoB**: Date of birth of the student.
     - **Gender**: M, f
     - **city**: The city where the student resides.
     - **street**: Address information of the student.
     - **phone**: Contact number for the student.
     - **email**: The student’s email address.
     - **password**: Login credential for the student.

2.**Instructor**

- Attributes:
  - **ins_id (Primary Key)**: A unique identifier for each instructor.
  - **ins_fname**: First name of the instructor.
  - **ins_lname**: Last name of the instructor.
  - **ins_DoB**: Date of birth of the instructor.
  - **city**: The city where the instructor resides.
  - **street**: Address information of the instructor.
  - **phone**: Contact number for the instructor.
  - **email**: The instructor’s email address.
  - **password**: Login credential for the instructor.
  - **salary**: Monthly salary of the instructor.
  - **hire_date**: The date the instructor joined the institution.
  - **Gender**: M, f

3. **Department**

Attributes:

1. **dept_id (Primary Key)**: A unique identifier for each department.
2. **dept_name**: Name of the department.
3. **dept_location**: Physical location of the department.
4. **phone**: Contact number of the department.
5. **mgr_hiredate**: Date when the manager started managing the department.

4. **Course**

- Attributes:

1. **crs_id (Primary Key)**: A unique identifier for each course.
2. **crs_name**: Name of the course.
3. **hours**: Number of hours required for the course.

5. **Topic**

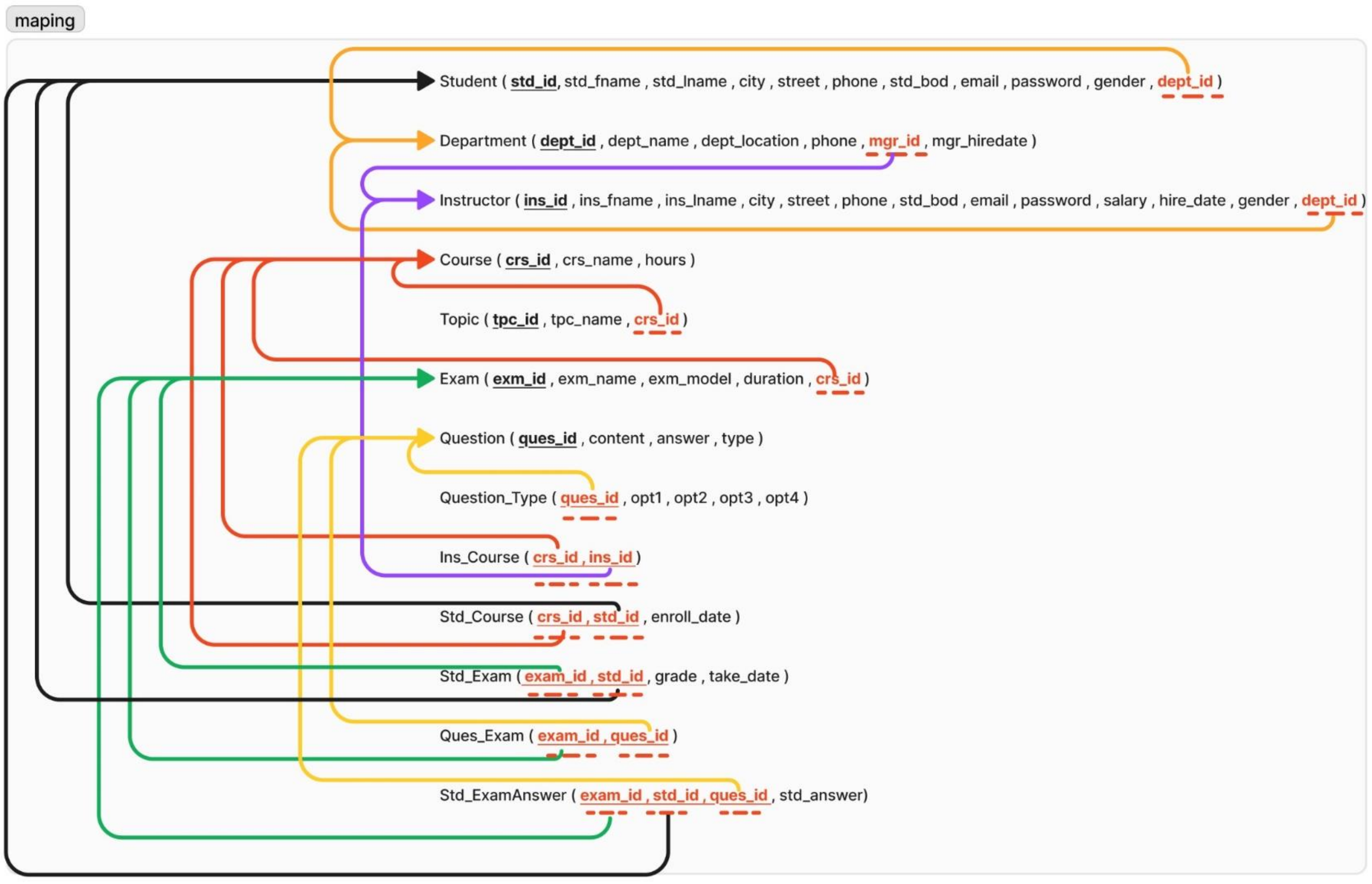
- Attributes:

1. **top_id (Primary Key)**: A unique identifier for each topic.
2. **top_name**: Name of the topic.

6. **Exam**
   - Attributes:
     1. **exam_id (Primary Key)**: A unique identifier for each exam.
     2. **exam_name**: Name of the exam.
     3. **exam_model**: Specifies whether the exam is multiple choice or theoretical.
     4. **duration**: Duration of the exam in minutes.

7. **Question** Attributes:
   1. **ques_id (Primary Key)**: A unique identifier for each question.
   2. **content**: The text of the question.
   3. **answer**: The correct answer for the question.
   4. **type**: Type of question (multiple choice, True or false).

➢ After Mapping :



Student (**std_id**, std_fname , std_lname , city , street , phone , std_bod , email , password , gender , dept_id)

Department (**dept_id** , dept_name , dept_location , phone , mgr_id , mgr_hiredate)

Instructor (**ins_id** , ins_fname , ins_lname , city , street , phone , std_bod , email , password , salary , hire_date , gender , dept_id)

Course (**crs_id** , crs_name , hours)

Topic (**tpc_id** , tpc_name , crs_id)

Exam (**exm_id** , exm_name , exm_model , duration , crs_id)

Question (**ques_id** , content , answer , type)

Question_Type (ques_id , opt1 , opt2 , opt3 , opt4)

Ins_Course (crs_id , ins_id)

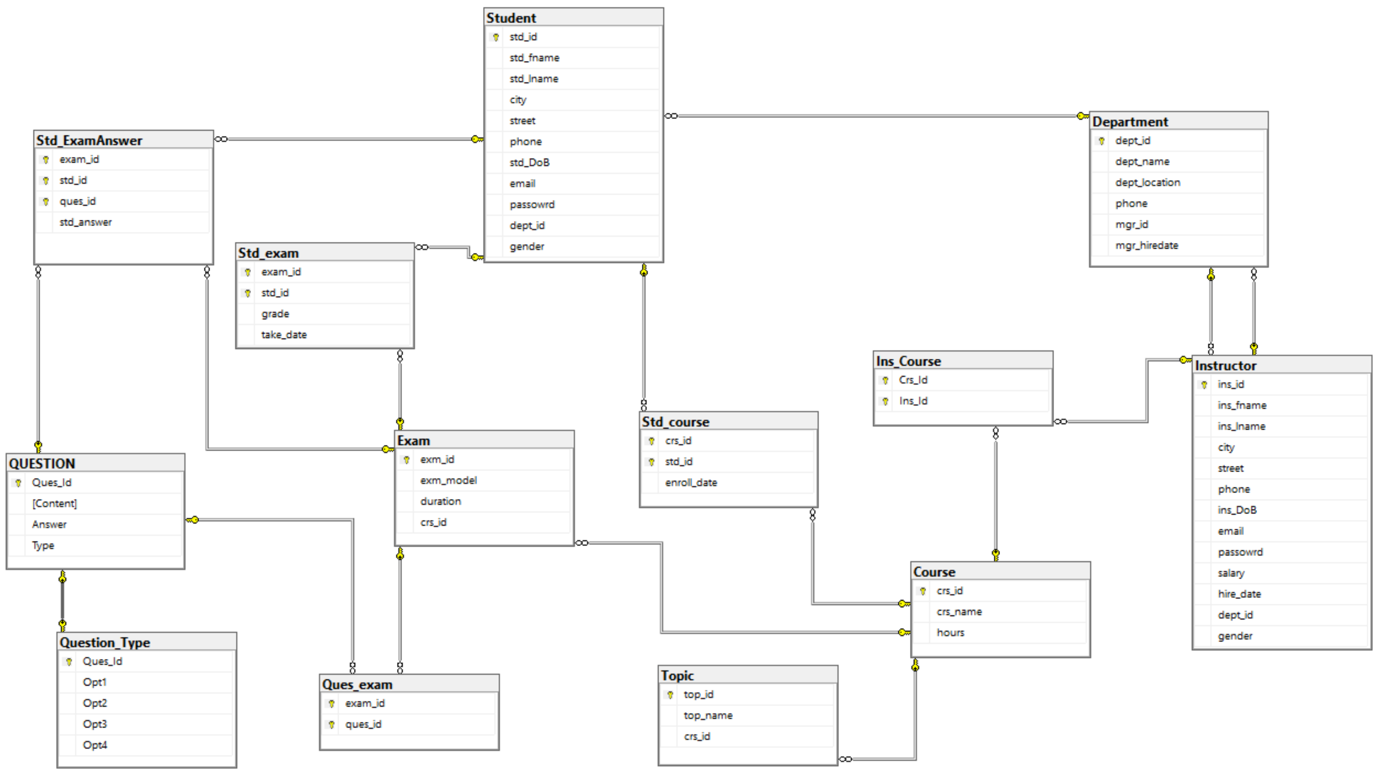
Std_Course (crs_id , std_id , enroll_date)

Std_Exam (exam_id , std_id , grade , take_date)

Ques_Exam (exam_id , ques_id)

Std_ExamAnswer (exam_id , std_id , ques_id , std_answer)

➢ We Retched to 13 Tables and 16 Relationships:



1. **Student**
    - **std_id (Primary Key)**: A unique identifier for each student.
    - **dept_id (Foreign Key)**: Links the student to their department.

2. **Instructor**
    - **ins_id (Primary Key)**: A unique identifier for each instructor.
    - **dept_id (Foreign Key)**: Links the instructor to their department.
3. **Department**
    - **mgr_id (Foreign Key)**: The instructor assigned as the manager.
    - **mgr_hiredate**: Date when the manager started managing the department.
4. **Course**
    - **crs_id (Primary Key)**: A unique identifier for each course.
5. **Topic**

- **top_id (Primary Key)**: A unique identifier for each topic.
- **crs_id (Foreign Key)**: Links the topic to a specific course.

6. **Exam**
   - **exam_id (Primary Key)**: A unique identifier for each exam.
   - **crs_id (Foreign Key)**: Links the exam to a specific course.
7. **Question**
   - **ques_id (Primary Key)**: A unique identifier for each question.
   - **type**: Type of question (e.g., multiple choice, theoretical).
8. **Question_Type**
   - **ques_id (Primary Key)**: A unique identifier for each question type.
   - **Opt1**, **Opt2**, **Opt3**, **Opt4**: The options for multiple-choice questions.
9. **Std_Course**
   - **crs_id (Foreign Key)**: References the course in which a student is enrolled.

- **std_id (Foreign Key)**: References the student enrolled in the course.
- **enroll_date**: The date the student enrolled in the course.

10. **Ins_Course**

    - **crs_id (Foreign Key)**: References the course in which an instructor supervises
    - **Ins_id (Foreign Key)**: References the instructor

11. **Std_Exam**

    - **exam_id (Foreign Key)**: Links the exam taken by the student.
    - **std_id (Foreign Key)**: Links the student who took the exam.
    - **grade**: The grade the student achieved in the exam.
    - **take_date**: The date the exam was taken.

12. **Std_ExamAnswer**

- **exam_id (Foreign Key)**: Links the exam the student answered.
- **std_id (Foreign Key)**: Links the student who answered the question.
- **ques_id (Foreign Key)**: Links the question that was answered.
- **std_answer**: The answer submitted by the student.

13. **Ques_Exam**

- **ques_id (Foreign Key)**: Links the question that was answered.
- **exam_id (Foreign Key)**: Links the exam the student answered.

Tables:

Tables:

Table dbo.Course *(24 rows)*

	Column	Data Type	Identity	Nullable	Default
PK	crs_id	int	X		
	crs_name	nvarchar(50)			
	hours	int			

Indexes:

PK__Course__ECAF5375790F450F *(Primary Key) (Clustered)*

crs_id

Referenced by:

dbo.Exam *(crs_id)*

dbo.Ins_Course *(Crs_Id -> crs_id)*

dbo.Std_course *(crs_id)*

dbo.Topic *(crs_id)*

```
CREATE TABLE [dbo].[Course](
    [crs_id] [int] IDENTITY(1,1) NOT NULL,
    [crs_name] [nvarchar](50) NOT NULL,
    [hours] [int] NOT NULL,
PRIMARY KEY CLUSTERED
(
    [crs_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table

dbo.Department *(12 rows)*

	Column	Data Type	Identity	Nullable	Default
PK	dept_id	int	X		
	dept_name	nvarchar(50)			
	dept_location	nvarchar(50)			
	phone	varchar(11)			
FK	mgr_id	int		X	
	mgr_hiredate	date		X	getdate()

Indexes:

PK__Departme__DCA659743C06EB4C *(Primary Key) (Clustered)*

dept_id

References:

dbo.Instructor *(mgr_id -> ins_id)*

Referenced by:

dbo.Instructor *(dept_id)*

dbo.Student *(dept_id)*

```sql
CREATE TABLE [dbo].[Department](
    [dept_id] [int] IDENTITY(1,1) NOT NULL,
    [dept_name] [nvarchar](50) NOT NULL,
    [dept_location] [nvarchar](50) NOT NULL,
    [phone] [varchar](11) NOT NULL,
    [mgr_id] [int] NULL,
    [mgr_hiredate] [date] NULL,
PRIMARY KEY CLUSTERED
(
    [dept_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
```

Table dbo.Exam *(4 rows)*

	Column	Data Type	Identity	Nullable	Default
PK	exm_id	int	X		
	exm_model	int			
	duration	int			
FK	crs_id	int		X	

Indexes:

PK__Exam__691115EC8D6620B1 *(Primary Key) (Clustered)*

exm_id

References:

dbo.Course *(crs_id)*

Referenced by:

dbo.Ques_exam *(exam_id -> exm_id)*

dbo.Std_exam *(exam_id -> exm_id)*

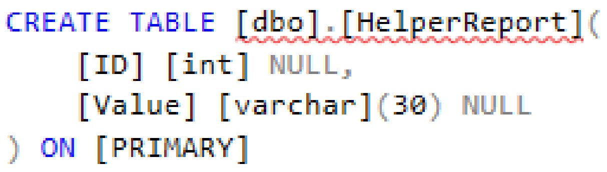
dbo.Std_ExamAnswer *(exam_id -> exm_id)*

```
CREATE TABLE [dbo].[Exam](
    [exm_id] [int] IDENTITY(1,1) NOT NULL,
    [exm_model] [int] NOT NULL,
    [duration] [int] NOT NULL,
    [crs_id] [int] NULL,
PRIMARY KEY CLUSTERED
(
    [exm_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
```

Table

dbo.HelperReport *(8 rows)*

	Column	Data Type	Identity	Nullable	Default
	ID	int		X	
	Value	varchar(30)		X	



```
CREATE TABLE [dbo].[HelperReport](
    [ID] [int] NULL,
    [Value] [varchar](30) NULL
) ON [PRIMARY]
```

Table

dbo.Ins_Course *(53 rows)*

	Column	Data Type	Identity	Nullable	Default
PK, FK	Crs_Id	int			
PK, FK	Ins_Id	int			

Indexes:

CI *(Primary Key) (Clustered)*

Ins_Id
Crs_Id

References:

dbo.Course *(Crs_Id -> crs_id)*

dbo.Instructor *(Ins_Id -> ins_id)*

```
CREATE TABLE [dbo].[Ins_Course](
    [Crs_Id] [int] NOT NULL,
    [Ins_Id] [int] NOT NULL,
 CONSTRAINT [CI] PRIMARY KEY CLUSTERED
(
    [Ins_Id] ASC,
    [Crs_Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
```

Table

dbo.Instructor *(32 rows)*

	Column	Data Type	Identity	Nullable	Default
PK	ins_id	int	X		
	ins_fname	nvarchar(20)			
	ins_lname	nvarchar(20)			
	city	nvarchar(20)			
	street	nvarchar(20)		X	
	phone	varchar(11)			
	ins_DoB	date			
UK	email	nvarchar(50)			
	passowrd	nvarchar(20)			
	salary	decimal(10,2)		X	10000
	hire_date	date		X	getdate()
FK	dept_id	int		X	
	gender	nvarchar(10)			'Not Specified'

Indexes:

PK__Instruct__9CB72D201E062DDE *(Primary Key) (Clustered)*

ins_id

```
CREATE TABLE [dbo].[Instructor](
    [ins_id] [int] IDENTITY(1,1) NOT NULL,
    [ins_fname] [nvarchar](20) NOT NULL,
    [ins_lname] [nvarchar](20) NOT NULL,
    [city] [nvarchar](20) NOT NULL,
    [street] [nvarchar](20) NULL,
    [phone] [varchar](11) NOT NULL,
    [ins_DoB] [date] NOT NULL,
    [email] [nvarchar](50) NOT NULL,
    [passowrd] [nvarchar](20) NOT NULL,
    [salary] [decimal](10, 2) NULL,
    [hire_date] [date] NULL,
    [dept_id] [int] NULL,
    [gender] [nvarchar](10) NOT NULL,
PRIMARY KEY CLUSTERED
(
    [ins_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
    [email] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table
UQ__Instruct__AB6E616466CE6C85 *(Unique)*

email

References:

dbo.Department *(dept_id)*

Referenced by:

dbo.Department *(mgr_id -> ins_id)*

dbo.Ins_Course *(Ins_Id -> ins_id)*

----

dbo.Ques_exam *(40 rows)*

	Column	Data Type	Identity	Nullable	Default
PK, FK	exam_id	int			
PK, FK	ques_id	int			

Indexes:

PK_Ques_exam *(Primary Key) (Clustered)*

exam_id ques_id

References: dbo.Exam *(exam_id -> exm_id)* **dbo.QUESTION** *(ques_id -> Ques_Id)*

```
CREATE TABLE [dbo].[Ques_exam](
    [exam_id] [int] NOT NULL,
    [ques_id] [int] NOT NULL,
 CONSTRAINT [PK_Ques_exam] PRIMARY KEY CLUSTERED
(
    [exam_id] ASC,
    [ques_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table
dbo.QUESTION *(102 rows)*

	Column	Data Type	Identity	Nullable	Default
PK	Ques_Id	int	X		
	Content	nvarchar(max)			
	Answer	nvarchar(max)			
	Type	varchar(20)			

Indexes:

PK__QUESTION__A821235E5B0D921A *(Primary Key) (Clustered)*

Ques_Id

Referenced by:

dbo.Ques_exam *(ques_id -> Ques_Id)*

dbo.Question_Type *(Ques_Id)*

dbo.Std_ExamAnswer *(ques_id -> Ques_Id)*

```
CREATE TABLE [dbo].[QUESTION](
    [Ques_Id] [int] IDENTITY(1,1) NOT NULL,
    [Content] [nvarchar](max) NOT NULL,
    [Answer] [nvarchar](max) NOT NULL,
    [Type] [varchar](20) NOT NULL,
PRIMARY KEY CLUSTERED
(
    [Ques_Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Table

dbo.Question_Type *(100 rows)*

	Column	Data Type	Identity	Nullable	Default
PK, FK	Ques_Id	int			
	Opt1	nvarchar(100)			
	Opt2	nvarchar(100)			
	Opt3	nvarchar(100)		X	
	Opt4	nvarchar(100)		X	

Indexes:

PK__Question__A821235E85B09D1A *(Primary Key) (Clustered)*

Ques_Id

References:

dbo.QUESTION *(Ques_Id)*

```
CREATE TABLE [dbo].[Question_Type](
    [Ques_Id] [int] NOT NULL,
    [Opt1] [nvarchar](100) NOT NULL,
    [Opt2] [nvarchar](100) NOT NULL,
    [Opt3] [nvarchar](100) NULL,
    [Opt4] [nvarchar](100) NULL,
PRIMARY KEY CLUSTERED
(
    [Ques_Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
```

Table
dbo.Std_course *(68 rows)*

	Column	Data Type	Identity	Nullable	Default
PK, FK	crs_id	int			
PK, FK	std_id	int			
	enroll_date	date			getdate()

Indexes:

PK_Std_course *(Primary Key) (Clustered)*

crs_id
std_id

References:

dbo.Course *(crs_id)*

dbo.Student *(std_id)*

```
CREATE TABLE [dbo].[Std_course](
    [crs_id] [int] NOT NULL,
    [std_id] [int] NOT NULL,
    [enroll_date] [date] NOT NULL,
 CONSTRAINT [PK_Std_course] PRIMARY KEY CLUSTERED
(
    [crs_id] ASC,
    [std_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table

dbo.Std_exam *(2 rows)*

	Column	Data Type	Identity	Nullable	Default
PK, FK	exam_id	int			
PK, FK	std_id	int			
	grade	int			70
	take_date	date			getdate()

Indexes:

PK_Std_exam *(Primary Key) (Clustered)*

exam_id
std_id

References: dbo.Exam *(exam_id ->*

exm_id)

dbo.Student *(std_id)*

```
CREATE TABLE [dbo].[Std_exam](
    [exam_id] [int] NOT NULL,
    [std_id] [int] NOT NULL,
    [grade] [int] NOT NULL,
    [take_date] [date] NOT NULL,
 CONSTRAINT [PK_Std_exam] PRIMARY KEY CLUSTERED
(
    [exam_id] ASC,
    [std_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table

dbo.Std_ExamAnswer *(20 rows)*

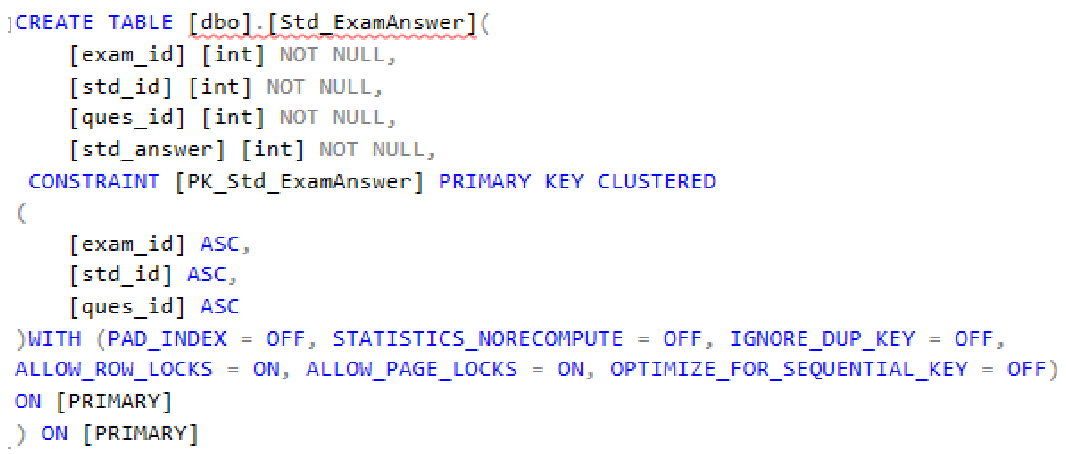
	Column	Data Type	Identity	Nullable	Default
PK, FK	exam_id	int			
PK, FK	std_id	int			
PK, FK	ques_id	int			
	std_answer	int			

Indexes:

PK_Std_ExamAnswer *(Primary Key) (Clustered)*

exam_id std_id ques_id

References: dbo.Exam *(exam_id -> exm_id)* **dbo.QUESTION** *(ques_id -> Ques_Id)* **dbo.Student** *(std_id)*



```
CREATE TABLE [dbo].[Std_ExamAnswer](
    [exam_id] [int] NOT NULL,
    [std_id] [int] NOT NULL,
    [ques_id] [int] NOT NULL,
    [std_answer] [int] NOT NULL,
 CONSTRAINT [PK_Std_ExamAnswer] PRIMARY KEY CLUSTERED
(
    [exam_id] ASC,
    [std_id] ASC,
    [ques_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
```

Table

dbo.Student *(68 rows)*

	Column	Data Type	Identity	Nullable	Default
PK	std_id	int	X		
	std_fname	nvarchar(20)			
	std_lname	nvarchar(20)			
	city	nvarchar(20)			
	street	nvarchar(50)		X	
	phone	varchar(11)			
	std_DoB	date			
UK	email	nvarchar(50)			
	passowrd	nvarchar(20)			
FK	dept_id	int		X	
	gender	nvarchar(3)			'M'

Indexes:

PK__Student__0B0245BA6AFBF50F *(Primary Key) (Clustered)*

std_id

UQ__Student__AB6E6164D4B7290A *(Unique)*

```
CREATE TABLE [dbo].[Student](
    [std_id] [int] IDENTITY(1,1) NOT NULL,
    [std_fname] [nvarchar](20) NOT NULL,
    [std_lname] [nvarchar](20) NOT NULL,
    [city] [nvarchar](20) NOT NULL,
    [street] [nvarchar](50) NULL,
    [phone] [varchar](11) NOT NULL,
    [std_DoB] [date] NOT NULL,
    [email] [nvarchar](50) NOT NULL,
    [passowrd] [nvarchar](20) NOT NULL,
    [dept_id] [int] NULL,
    [gender] [nvarchar](3) NOT NULL,
PRIMARY KEY CLUSTERED
(
    [std_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
    [email] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Table

email

References:

dbo.Department *(dept_id)*

Referenced by:

dbo.Std_course *(std_id)*

dbo.Std_exam *(std_id)*

dbo.Std_ExamAnswer *(std_id)*

---

dbo.Topic *(84 rows)*

	Column	Data Type	Identity	Nullable	Default
PK	top_id	int	X		
	top_name	nvarchar(50)			
FK	crs_id	int		X	

Indexes:

PK__Topic__B582A63DC71A6B3B *(Primary Key) (Clustered)*

top_id

References:

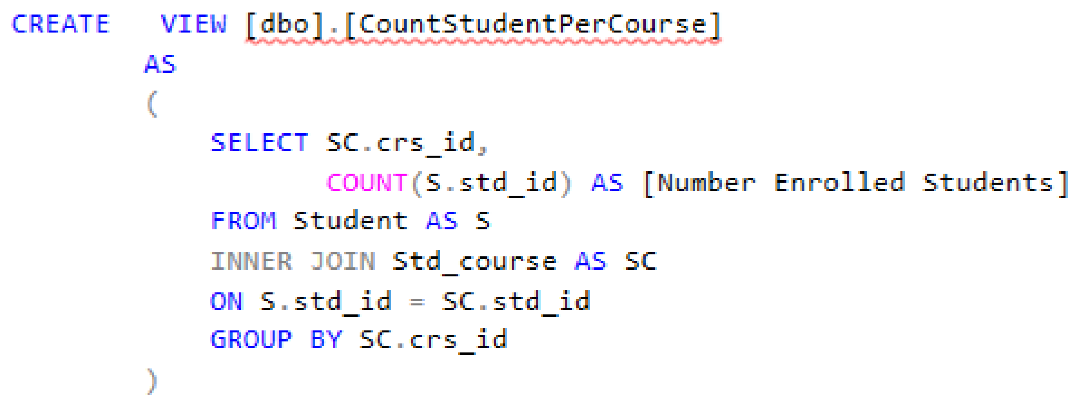
dbo.Course *(crs_id)*

```
CREATE TABLE [dbo].[Topic](
    [top_id] [int] IDENTITY(1,1) NOT NULL,
    [top_name] [nvarchar](50) NOT NULL,
    [crs_id] [int] NULL,
PRIMARY KEY CLUSTERED
(
    [top_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Views:

View dbo.CountStudentPerCourse

Column	Data Type	Nullable
crs_id	int	
Number Enrolled Students	int	X



```sql
CREATE   VIEW [dbo].[CountStudentPerCourse]
        AS
        (
            SELECT SC.crs_id,
                   COUNT(S.std_id) AS [Number Enrolled Students]
            FROM Student AS S
            INNER JOIN Std_course AS SC
            ON S.std_id = SC.std_id
            GROUP BY SC.crs_id
        )
```

Procedures:

Procedure dbo.AddCourse

Parameter	Data Type	Default	Is Output
@crs_name	nvarchar(50)		
@hours	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(16)	

```sql
CREATE   PROCEDURE [dbo].[AddCourse](@crs_name nvarchar(50), @hours int)
AS
    BEGIN TRY
        INSERT INTO Course
        VALUES(@crs_name, @hours);
    END TRY
    BEGIN CATCH
        SELECT 'Insertion Failed' AS [Error Message];
    END CATCH;
```

Procedure

dbo.AddDepartment

Parameter	Data Type	Default	Is Output
@dept_name	nvarchar(50)		
@dept_location	nvarchar(50)		
@phone	varchar(11)		
@mgr_id	int		
@mgr_hiredate	date		

Result:

Column	Data Type	Nullable
Error Message	varchar(16)	

```
CREATE   PROCEDURE [dbo].[AddDepartment]
(@dept_name nvarchar(50),
 @dept_location nvarchar(50),
 @phone varchar(11),
 @mgr_id int, @mgr_hiredate date)
AS
    BEGIN TRY
        INSERT INTO Department
        VALUES(@dept_name, @dept_location, @phone, @mgr_id, @mgr_hiredate);
    END TRY
    BEGIN CATCH
        SELECT 'Insertion Failed' AS [Error Message];
    END CATCH;
```

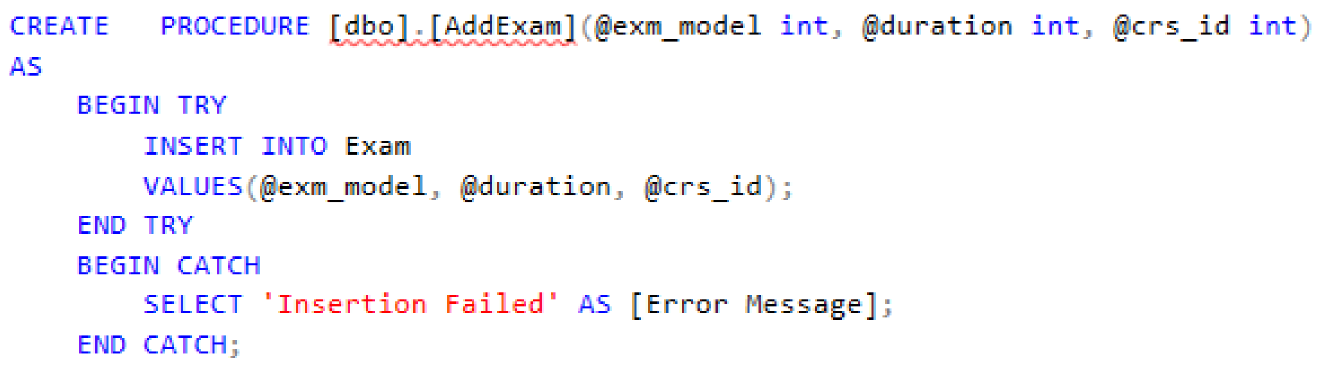
Procedure

dbo.AddExam

Parameter	Data Type	Default	Is Output
@exm_model	int		
@duration	int		
@crs_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(16)	



```sql
CREATE   PROCEDURE [dbo].[AddExam](@exm_model int, @duration int, @crs_id int)
AS
    BEGIN TRY
        INSERT INTO Exam
        VALUES(@exm_model, @duration, @crs_id);
    END TRY
    BEGIN CATCH
        SELECT 'Insertion Failed' AS [Error Message];
    END CATCH;
```

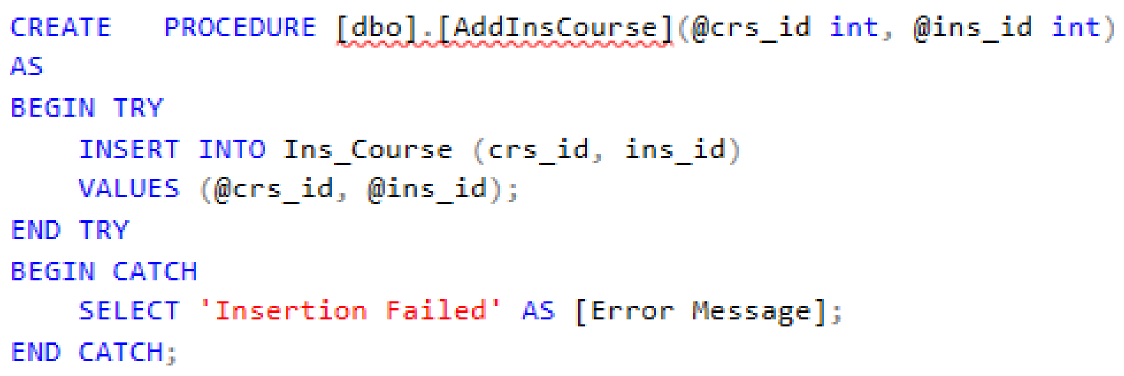
Procedure

dbo.AddInsCourse

Parameter	Data Type	Default	Is Output
@crs_id	int		
@ins_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(16)	



```
CREATE   PROCEDURE [dbo].[AddInsCourse](@crs_id int, @ins_id int)
AS
BEGIN TRY
    INSERT INTO Ins_Course (crs_id, ins_id)
    VALUES (@crs_id, @ins_id);
END TRY
BEGIN CATCH
    SELECT 'Insertion Failed' AS [Error Message];
END CATCH;
```

Procedure

dbo.AddInstructor

Parameter	Data Type	Default	Is Output
@ins_fname	nvarchar(20)		
@ins_lname	nvarchar(20)		
@city	nvarchar(20)		
@street	nvarchar(50)		
@phone	varchar(11)		
@ins_DoB	date		
@email	nvarchar(50)		
@passowrd	nvarchar(20)		
@salary	decimal(10,2)		
@hire_date	date		
@dept_id	int		
@gneder	nvarchar(10)		

Result:

Column	Data Type	Nullable
Error Message	varchar(16)	

```
CREATE   PROCEDURE [dbo].[AddInstructor]
( @ins_fname nvarchar(20), @ins_lname nvarchar(20),
  @city nvarchar(20), @street nvarchar(50),
  @phone varchar(11), @ins_DoB date,
  @email nvarchar(50), @passowrd nvarchar(20),
  @salary decimal(10, 2), @hire_date date,
  @dept_id int,@gneder nvarchar(10) )
AS
    BEGIN TRY
        INSERT INTO Instructor
        VALUES(@ins_fname, @ins_lname, @city,
               @street, @phone, @ins_DoB, @email,
               @passowrd, @salary, @hire_date, @dept_id,@gneder);
    END TRY
    BEGIN CATCH
        SELECT 'Insertion Failed' AS [Error Message];
    END CATCH;
```

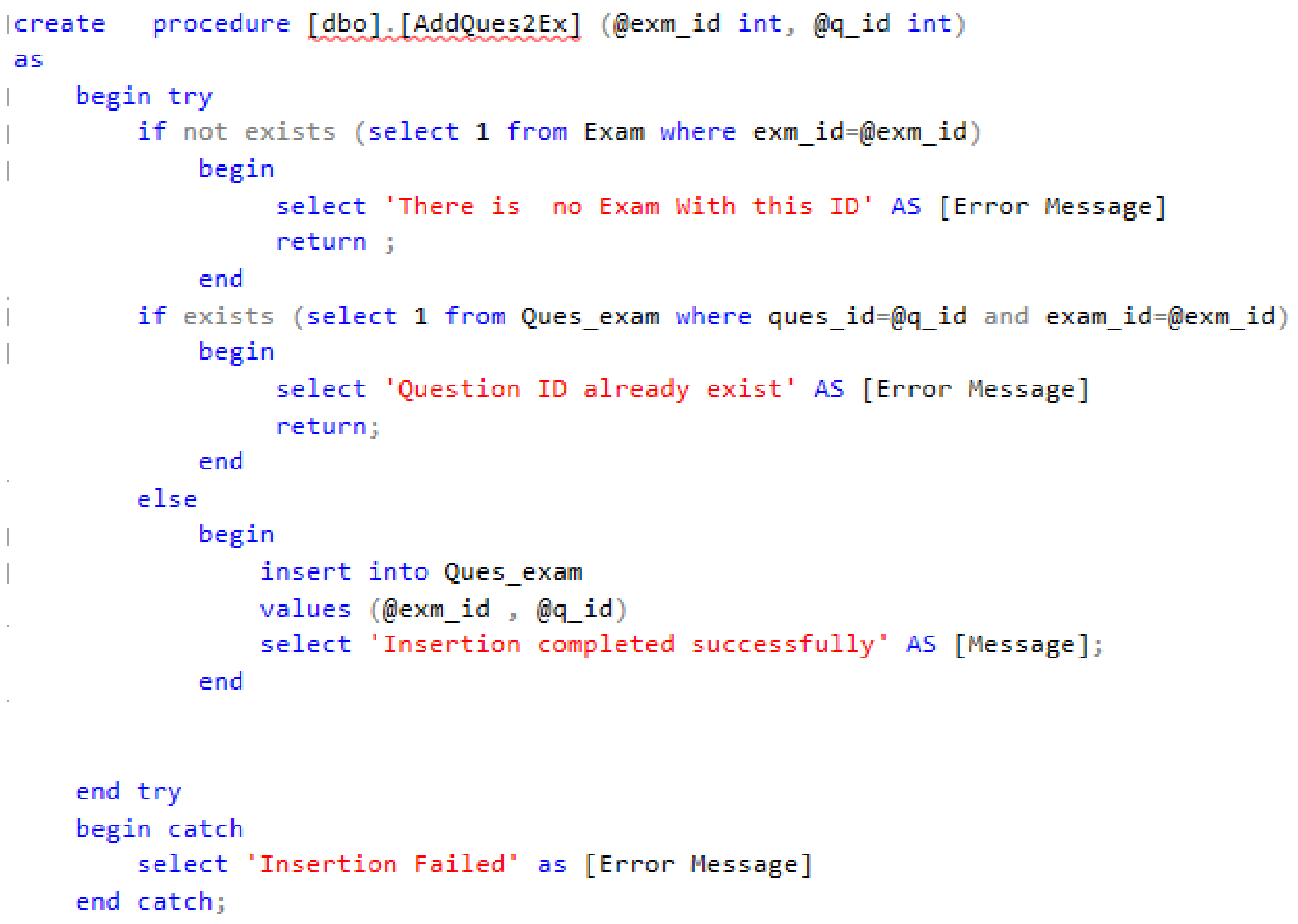
Procedure

dbo.AddQues2Ex

Parameter	Data Type	Default	Is Output
@exm_id	int		
@q_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(30)	



```
create   procedure [dbo].[AddQues2Ex] (@exm_id int, @q_id int)
as
    begin try
        if not exists (select 1 from Exam where exm_id=@exm_id)
            begin
                 select 'There is  no Exam With this ID' AS [Error Message]
                 return ;
            end
        if exists (select 1 from Ques_exam where ques_id=@q_id and exam_id=@exm_id)
            begin
                 select 'Question ID already exist' AS [Error Message]
                 return;
            end
        else
            begin
                insert into Ques_exam
                values (@exm_id , @q_id)
                select 'Insertion completed successfully' AS [Message];
            end


    end try
    begin catch
        select 'Insertion Failed' as [Error Message]
    end catch;
```

Procedure

dbo.AddQUESTION

Parameter	Data Type	Default	Is Output
@Content	nvarchar(max)		
@Answer	nvarchar(max)		
@Type	varchar(20)		

Result:

Column	Data Type	Nullable
Error Message	varchar(16)	

```
CREATE   PROCEDURE [dbo].[AddQUESTION]
( @Content nvarchar(max),
  @Answer nvarchar(max),
  @Type varchar(20)
)
AS
    BEGIN TRY
        INSERT INTO QUESTION
        VALUES(@Content, @Answer, @Type);
    END TRY
    BEGIN CATCH
        SELECT 'Insertion Failed' AS [Error Message];
    END CATCH;
```

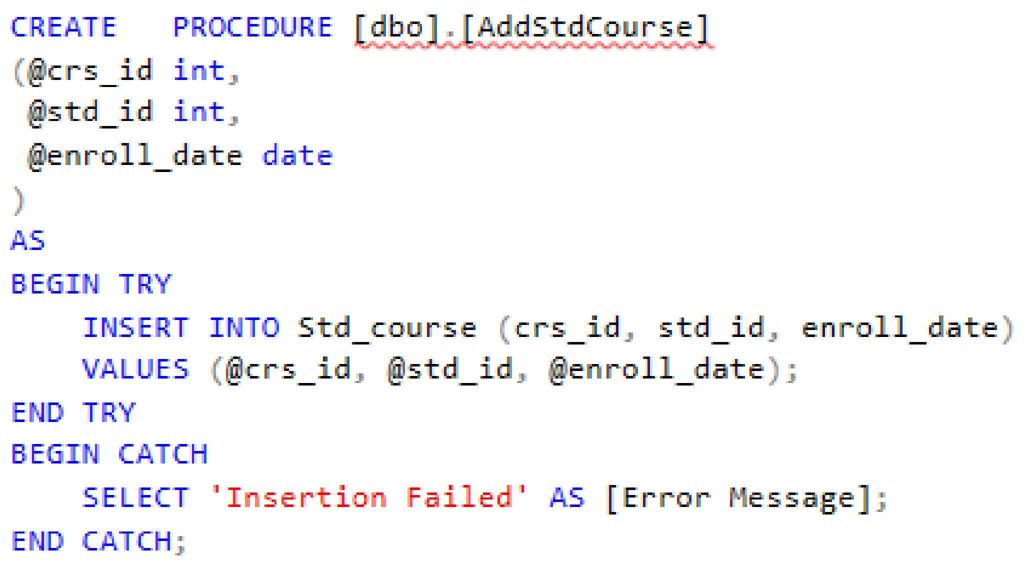
Procedure

dbo.AddStdCourse

Parameter	Data Type	Default	Is Output
@crs_id	int		
@std_id	int		
@enroll_date	date		

Result:

Column	Data Type	Nullable
Error Message	varchar(16)	



```sql
CREATE   PROCEDURE [dbo].[AddStdCourse]
(@crs_id int,
 @std_id int,
 @enroll_date date
)
AS
BEGIN TRY
    INSERT INTO Std_course (crs_id, std_id, enroll_date)
    VALUES (@crs_id, @std_id, @enroll_date);
END TRY
BEGIN CATCH
    SELECT 'Insertion Failed' AS [Error Message];
END CATCH;
```

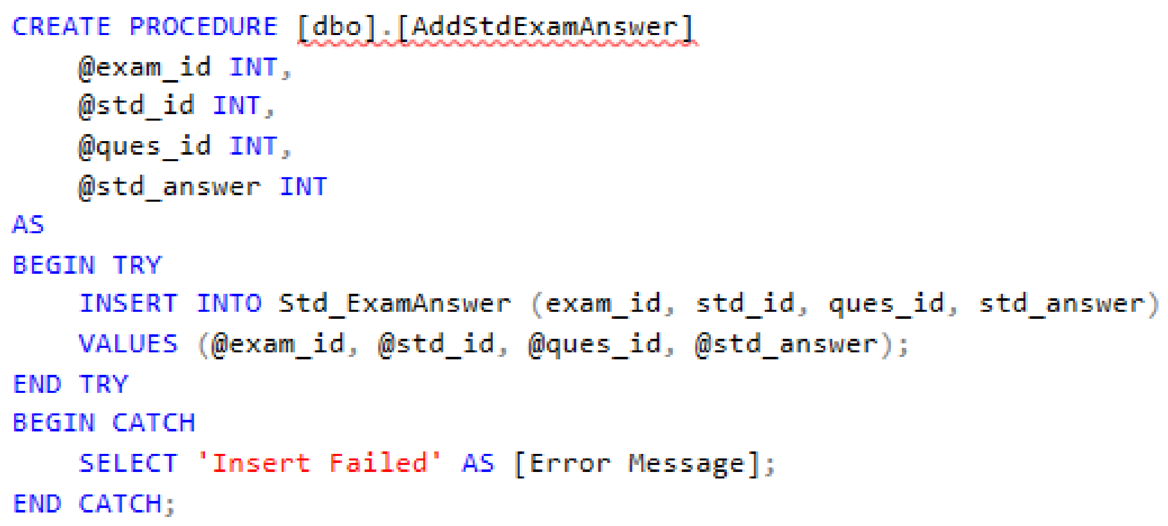
Procedure

dbo.AddStdExamAnswer

Parameter	Data Type	Default	Is Output
@exam_id	int		
@std_id	int		
@ques_id	int		
@std_answer	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```sql
CREATE PROCEDURE [dbo].[AddStdExamAnswer]
    @exam_id INT,
    @std_id INT,
    @ques_id INT,
    @std_answer INT
AS
BEGIN TRY
    INSERT INTO Std_ExamAnswer (exam_id, std_id, ques_id, std_answer)
    VALUES (@exam_id, @std_id, @ques_id, @std_answer);
END TRY
BEGIN CATCH
    SELECT 'Insert Failed' AS [Error Message];
END CATCH;
```

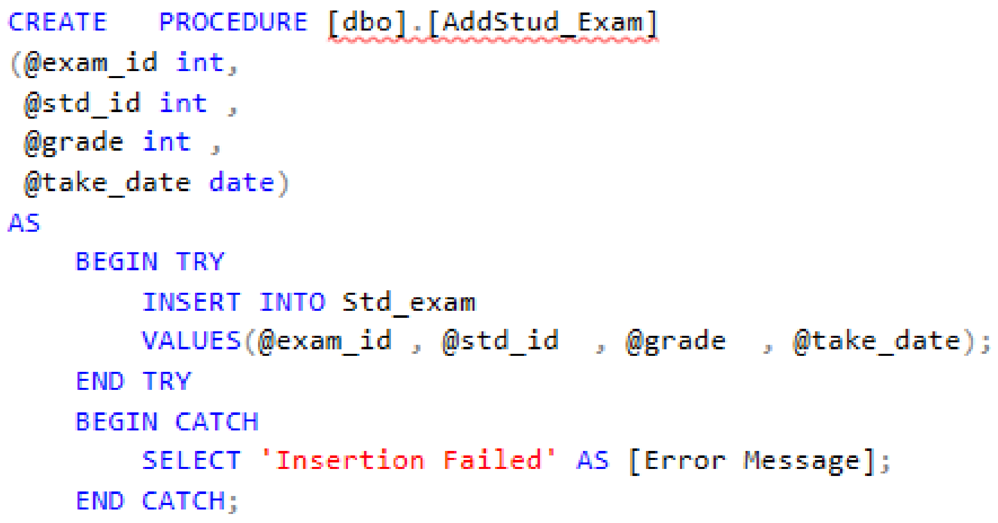
Procedure

dbo.AddStud_Exam

Parameter	Data Type	Default	Is Output
@exam_id	int		
@std_id	int		
@grade	int		
@take_date	date		

Result:

Column	Data Type	Nullable
Error Message	varchar(16)	



```
CREATE   PROCEDURE [dbo].[AddStud_Exam]
(@exam_id int,
 @std_id int ,
 @grade int ,
 @take_date date)
AS
    BEGIN TRY
        INSERT INTO Std_exam
        VALUES(@exam_id , @std_id  , @grade  , @take_date);
    END TRY
    BEGIN CATCH
        SELECT 'Insertion Failed' AS [Error Message];
    END CATCH;
```

Procedure

dbo.AddStudent

Parameter	Data Type	Default	Is Output
@std_fname	nvarchar(20)		
@std_lname	nvarchar(20)		
@city	nvarchar(20)		
@street	nvarchar(50)		
@phone	varchar(11)		
@std_DoB	date		
@email	nvarchar(50)		
@passowrd	nvarchar(20)		
@dept_id	int		
@gneder	nvarchar(10)		

Result:

Column	Data Type	Nullable
Error Message	nvarchar(4000)	X

```
CREATE   PROCEDURE [dbo].[AddStudent]
(@std_fname nvarchar(20), @std_lname nvarchar(20), @city nvarchar(20),
 @street nvarchar(50), @phone varchar(11), @std_DoB date, @email nvarchar(50),
 @passowrd nvarchar(20), @dept_id int , @gneder nvarchar(10))
AS
    BEGIN TRY
        INSERT INTO Student
        VALUES(@std_fname, @std_lname, @city, @street, @phone, @std_DoB,
               @email, @passowrd, @dept_id, @gneder);
    END TRY
    BEGIN CATCH
        SELECT ERROR_MESSAGE() AS [Error Message];
    END CATCH;
```

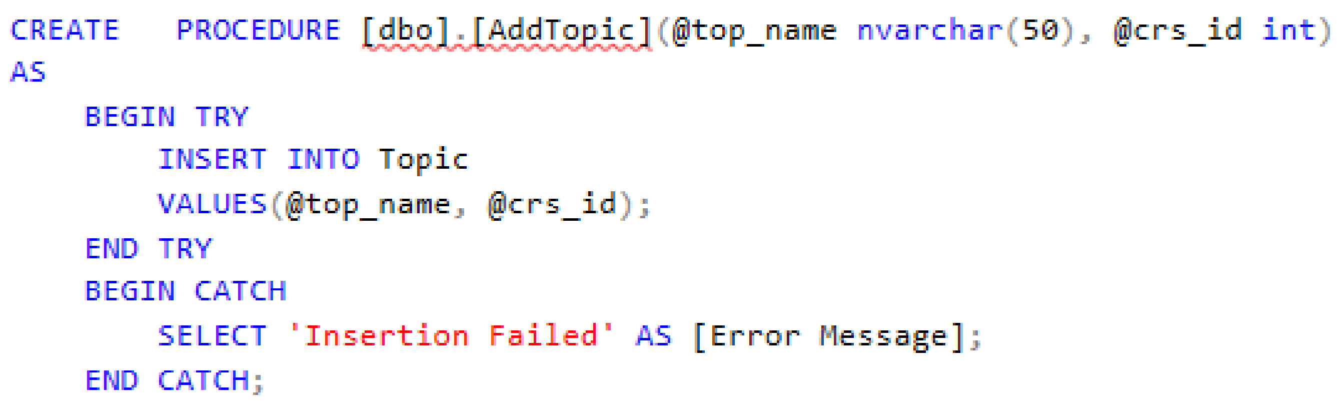
Procedure

dbo.AddTopic

Parameter	Data Type	Default	Is Output
@top_name	nvarchar(50)		
@crs_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(16)	



```
CREATE   PROCEDURE [dbo].[AddTopic](@top_name nvarchar(50), @crs_id int)
AS
    BEGIN TRY
        INSERT INTO Topic
        VALUES(@top_name, @crs_id);
    END TRY
    BEGIN CATCH
        SELECT 'Insertion Failed' AS [Error Message];
    END CATCH;
```

Procedure

dbo.CalculateGrade

Parameter	Data Type	Default	Is Output
@exam_id	int		
@student_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(33)	

```
CREATE   PROCEDURE [dbo].[CalculateGrade](@exam_id int, @student_id int)
AS
    BEGIN TRY
        DECLARE @NumberQuestion INT = (SELECT COUNT(*) FROM Ques_exam WHERE exam_id = @exam_id);
        DECLARE @TotalDegree INT;

        SET @TotalDegree =
        (
            SELECT SUM
                    (
                        CASE
                             WHEN (Q.Answer =  QP.Answer) THEN 1
                             ELSE 0
                        END
                    ) AS TotalDegree
            FROM GetExamOption(@exam_id, @student_id) AS QP
            INNER JOIN QUESTION Q
            ON QP.ques_id = Q.ques_id
        );
        DECLARE @DegreePercantge INT = (SELECT @TotalDegree / (1.0 * @NumberQuestion) * 100)

        INSERT INTO Std_exam
        VALUES(@exam_id, @student_id, @DegreePercantge, '2023-06-07')
    END TRY

    BEGIN CATCH
        SELECT 'Can''t Calculate Grade OF STUDENT!' AS [Error Message];
    END CATCH
```

Procedure

dbo.CreateExam

Parameter	Data Type	Default	Is Output
@crs_id	int		
@exam_model	int		
@duration	int		
@quesTureFalse	int		
@quesMCQ	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(22)	

```
CREATE  PROCEDURE [dbo].[CreateExam]
(@crs_id int, @exam_model int,
 @duration int, @quesTureFalse int, @quesMCQ int)
AS
    BEGIN TRY
        DECLARE @newExamId INT;
        INSERT INTO Exam
        VALUES(@exam_model, @duration, @crs_id);
         SET @newExamId = (SELECT exm_id FROM Exam WHERE crs_id = @crs_id);
    END TRY

    BEGIN CATCH
        SELECT 'Can''t Insert New Exam!' AS [Error Message];
    END CATCH

    BEGIN TRY
        INSERT INTO Ques_exam
        SELECT TOP (@quesTureFalse) @newExamId,
               Ques_Id
        FROM QUESTION
        WHERE Type = 'True&False'
        ORDER BY NEWID();

        INSERT INTO Ques_exam
        SELECT TOP(@quesMCQ) @newExamId,
               Ques_Id
        FROM QUESTION
        WHERE Type = 'MCQ'
        ORDER BY NEWID();
    END TRY

    BEGIN CATCH
        SELECT 'Can''t Insert New Exam!' AS [Error Message];
    END CATCH
```

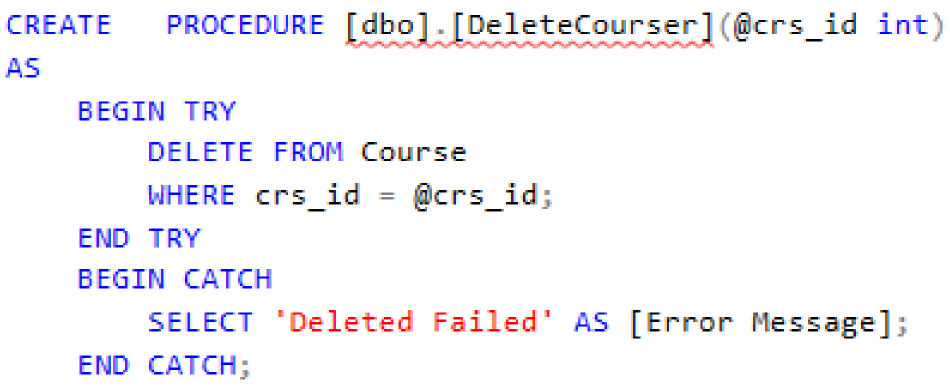
Procedure

dbo.DeleteCourser

Parameter	Data Type	Default	Is Output
@crs_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(14)	



```
CREATE   PROCEDURE [dbo].[DeleteCourser](@crs_id int)
AS
    BEGIN TRY
        DELETE FROM Course
        WHERE crs_id = @crs_id;
    END TRY
    BEGIN CATCH
        SELECT 'Deleted Failed' AS [Error Message];
    END CATCH;
```

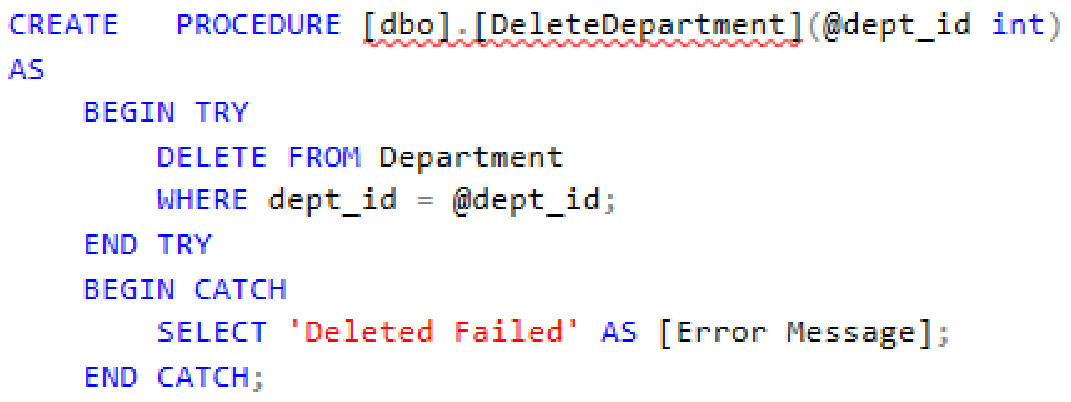
Procedure

dbo.DeleteDepartment

Parameter	Data Type	Default	Is Output
@dept_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(14)	



```
CREATE   PROCEDURE [dbo].[DeleteDepartment](@dept_id int)
AS
    BEGIN TRY
        DELETE FROM Department
        WHERE dept_id = @dept_id;
    END TRY
    BEGIN CATCH
        SELECT 'Deleted Failed' AS [Error Message];
    END CATCH;
```

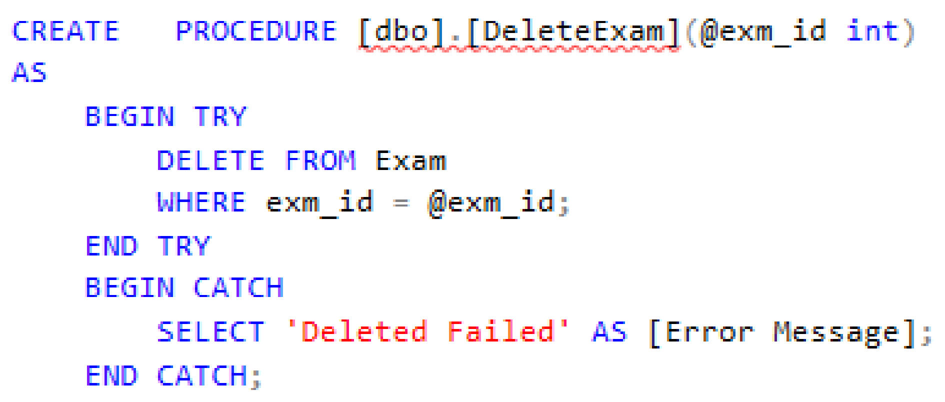
Procedure

dbo.DeleteExam

Parameter	Data Type	Default	Is Output
@exm_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(14)	



```
CREATE   PROCEDURE [dbo].[DeleteExam](@exm_id int)
AS
    BEGIN TRY
        DELETE FROM Exam
        WHERE exm_id = @exm_id;
    END TRY
    BEGIN CATCH
        SELECT 'Deleted Failed' AS [Error Message];
    END CATCH;
```

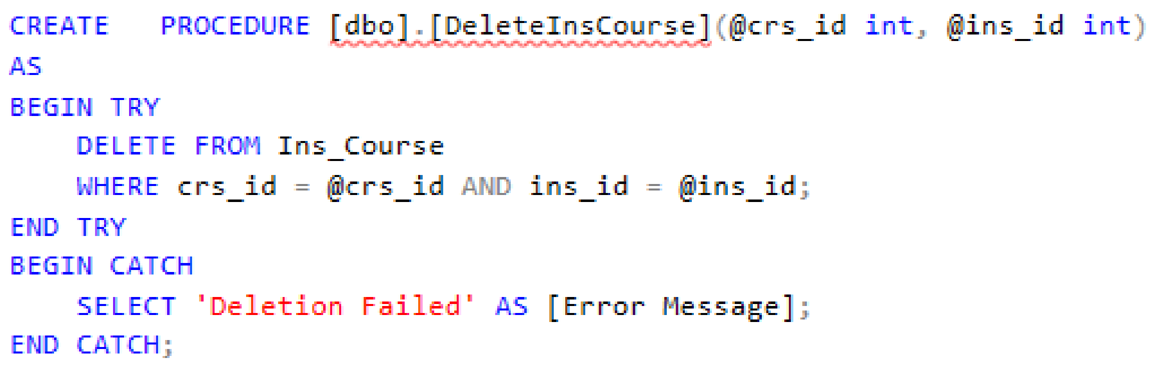
Procedure

dbo.DeleteInsCourse

Parameter	Data Type	Default	Is Output
@crs_id	int		
@ins_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(15)	



```
CREATE   PROCEDURE [dbo].[DeleteInsCourse](@crs_id int, @ins_id int)
AS
BEGIN TRY
    DELETE FROM Ins_Course
    WHERE crs_id = @crs_id AND ins_id = @ins_id;
END TRY
BEGIN CATCH
    SELECT 'Deletion Failed' AS [Error Message];
END CATCH;
```

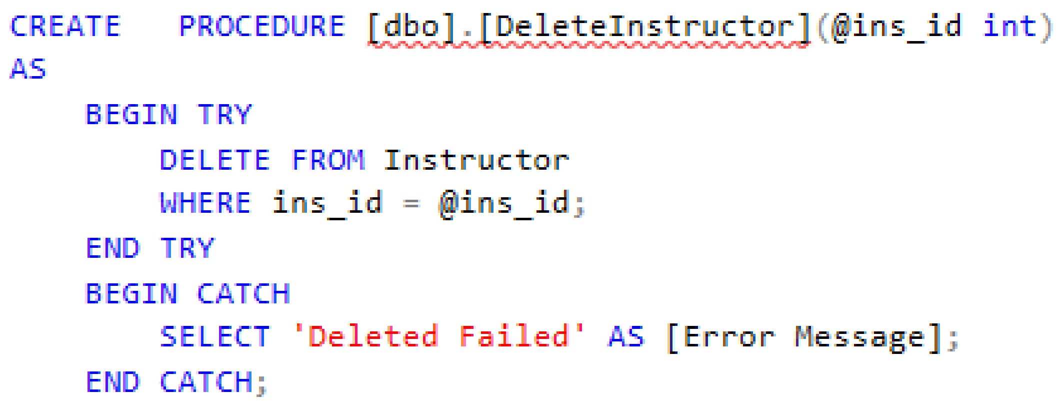
Procedure

dbo.DeleteInstructor

Parameter	Data Type	Default	Is Output
@ins_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(14)	



```
CREATE   PROCEDURE [dbo].[DeleteInstructor](@ins_id int)
AS
    BEGIN TRY
        DELETE FROM Instructor
        WHERE ins_id = @ins_id;
    END TRY
    BEGIN CATCH
        SELECT 'Deleted Failed' AS [Error Message];
    END CATCH;
```

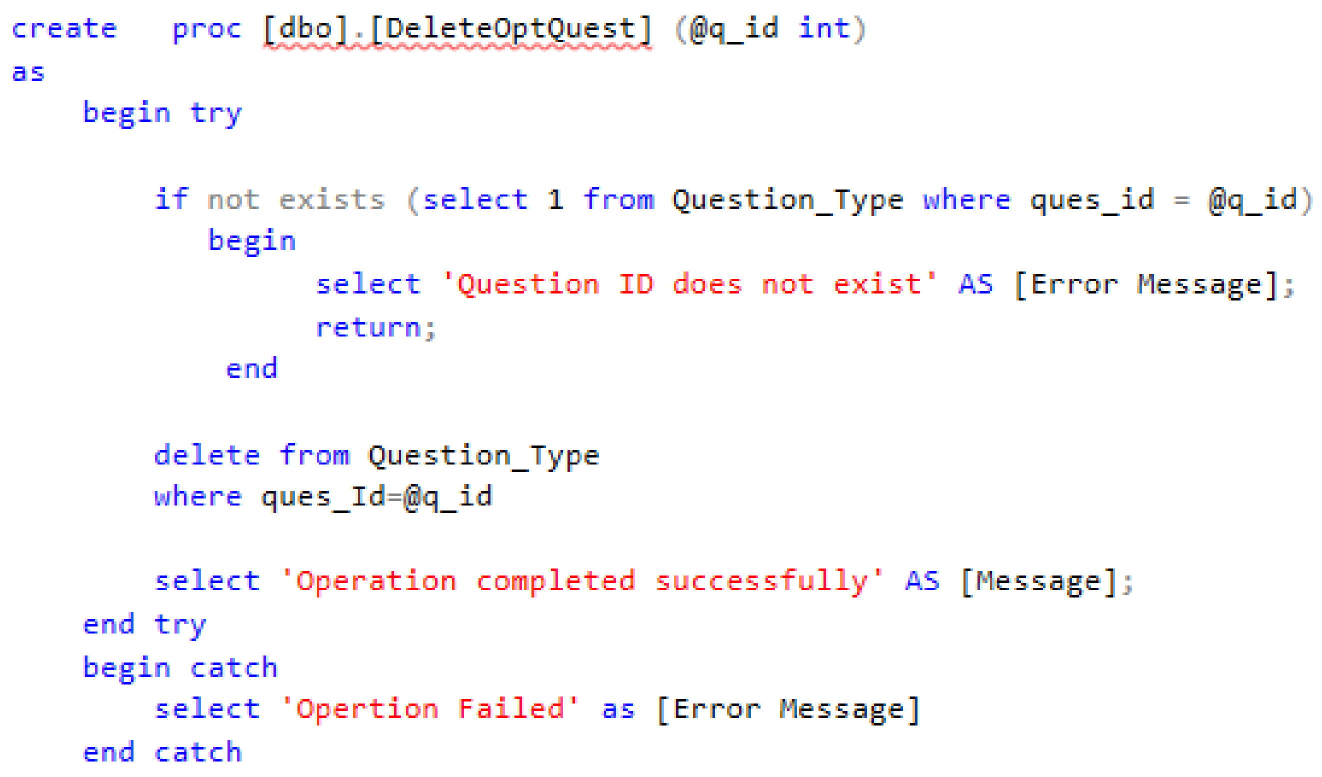
Procedure

dbo.DeleteOptQuest

Parameter	Data Type	Default	Is Output
@q_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(26)	



```
create   proc [dbo].[DeleteOptQuest] (@q_id int)
as
    begin try

        if not exists (select 1 from Question_Type where ques_id = @q_id)
           begin
                 select 'Question ID does not exist' AS [Error Message];
                 return;
            end

        delete from Question_Type
        where ques_Id=@q_id

        select 'Operation completed successfully' AS [Message];
    end try
    begin catch
        select 'Opertion Failed' as [Error Message]
    end catch
```

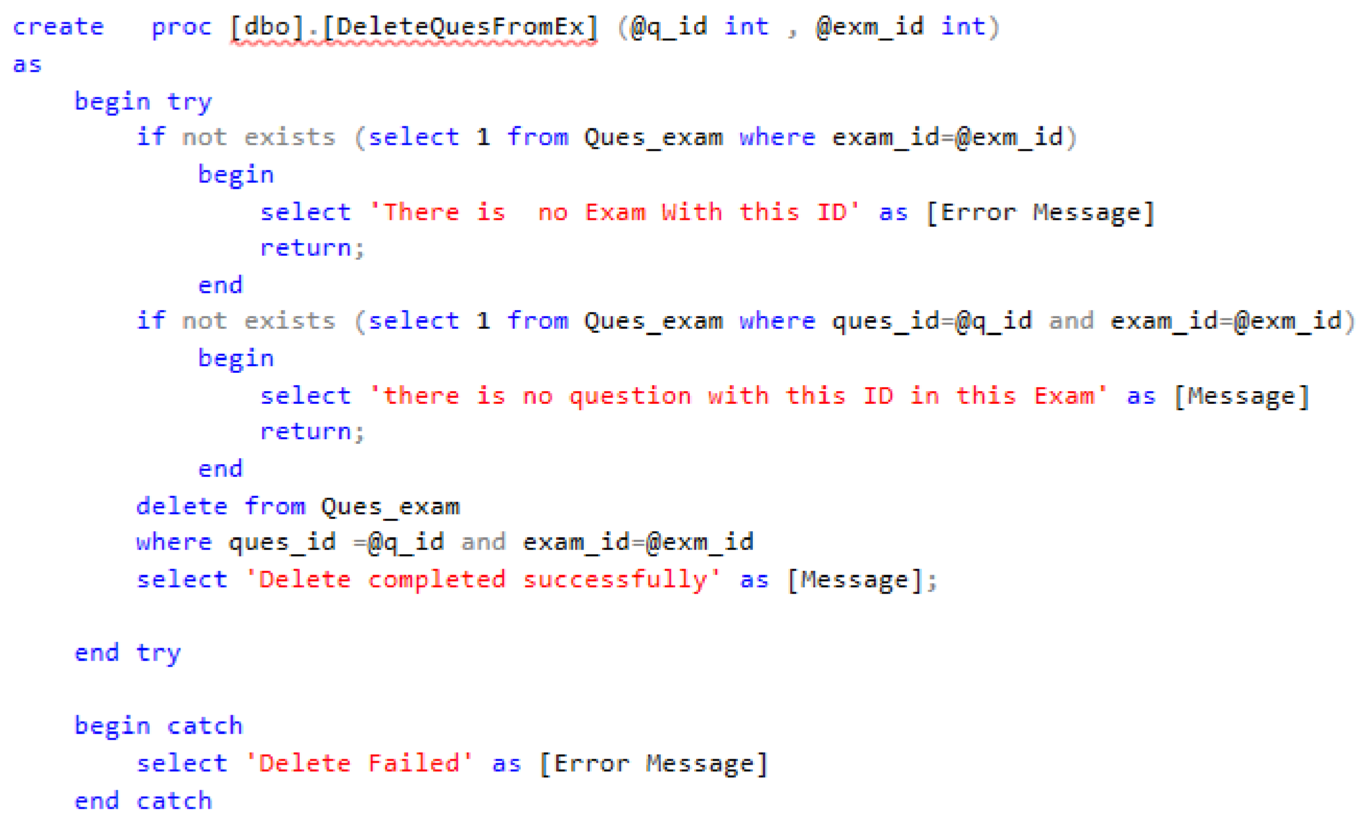
Procedure

dbo.DeleteQuesFromEx

Parameter	Data Type	Default	Is Output
@q_id	int		
@exm_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(30)	



```sql
create   proc [dbo].[DeleteQuesFromEx] (@q_id int , @exm_id int)
as
    begin try
        if not exists (select 1 from Ques_exam where exam_id=@exm_id)
            begin
                select 'There is  no Exam With this ID' as [Error Message]
                return;
            end
        if not exists (select 1 from Ques_exam where ques_id=@q_id and exam_id=@exm_id)
            begin
                select 'there is no question with this ID in this Exam' as [Message]
                return;
            end
        delete from Ques_exam
        where ques_id =@q_id and exam_id=@exm_id
        select 'Delete completed successfully' as [Message];

    end try

    begin catch
        select 'Delete Failed' as [Error Message]
    end catch
```

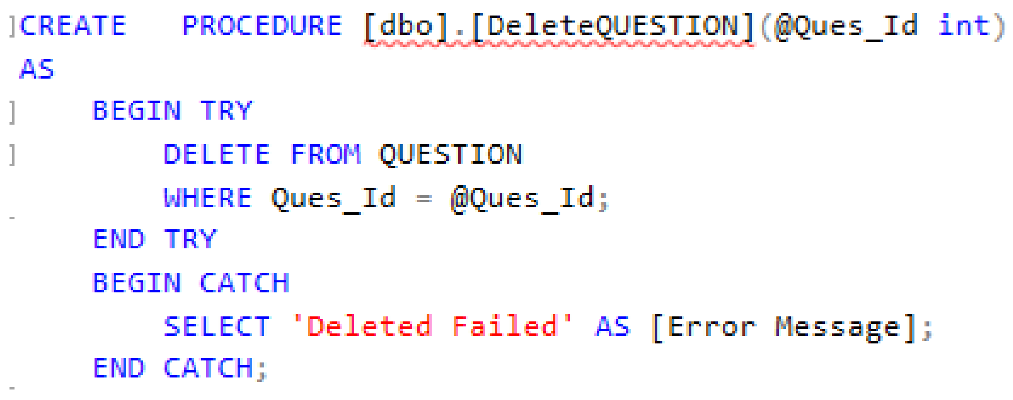
Procedure

dbo.DeleteQUESTION

Parameter	Data Type	Default	Is Output
@Ques_Id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(14)	



```
CREATE   PROCEDURE [dbo].[DeleteQUESTION](@Ques_Id int)
AS
    BEGIN TRY
        DELETE FROM QUESTION
        WHERE Ques_Id = @Ques_Id;
    END TRY
    BEGIN CATCH
        SELECT 'Deleted Failed' AS [Error Message];
    END CATCH;
```

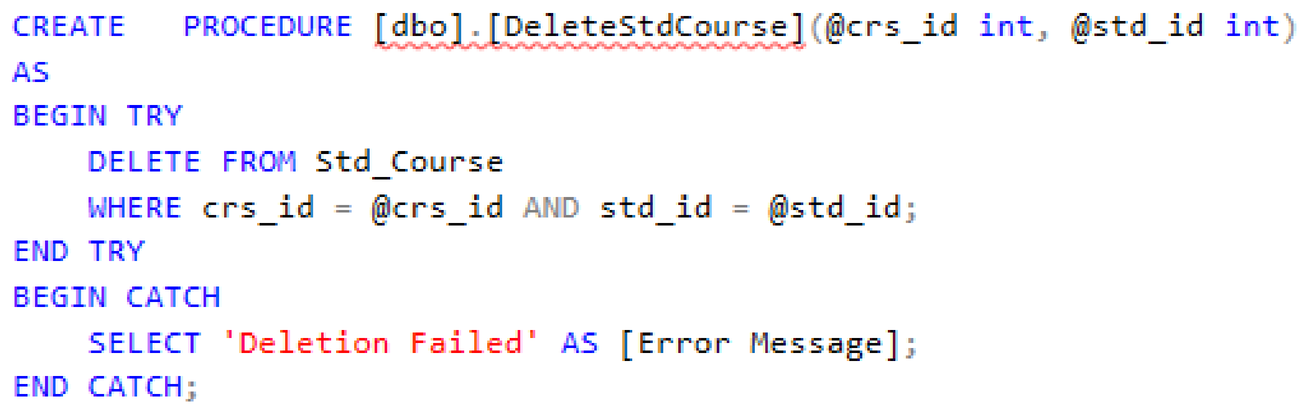
Procedure

dbo.DeleteStdCourse

Parameter	Data Type	Default	Is Output
@crs_id	int		
@std_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(15)	



```sql
CREATE   PROCEDURE [dbo].[DeleteStdCourse](@crs_id int, @std_id int)
AS
BEGIN TRY
    DELETE FROM Std_Course
    WHERE crs_id = @crs_id AND std_id = @std_id;
END TRY
BEGIN CATCH
    SELECT 'Deletion Failed' AS [Error Message];
END CATCH;
```

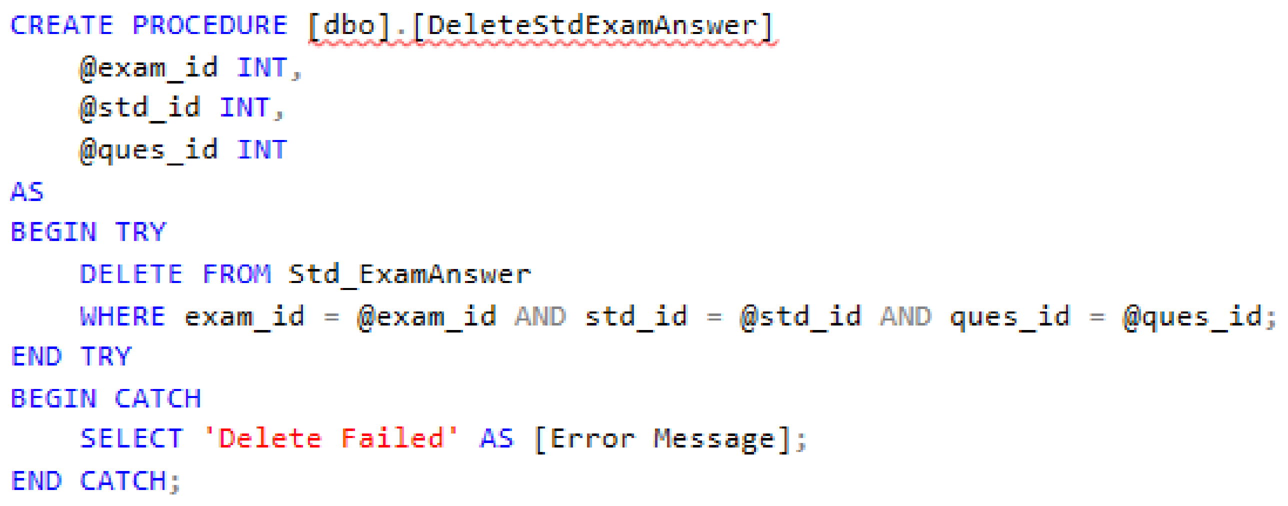
Procedure

dbo.DeleteStdExamAnswer

Parameter	Data Type	Default	Is Output
@exam_id	int		
@std_id	int		
@ques_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```sql
CREATE PROCEDURE [dbo].[DeleteStdExamAnswer]
    @exam_id INT,
    @std_id INT,
    @ques_id INT
AS
BEGIN TRY
    DELETE FROM Std_ExamAnswer
    WHERE exam_id = @exam_id AND std_id = @std_id AND ques_id = @ques_id;
END TRY
BEGIN CATCH
    SELECT 'Delete Failed' AS [Error Message];
END CATCH;
```

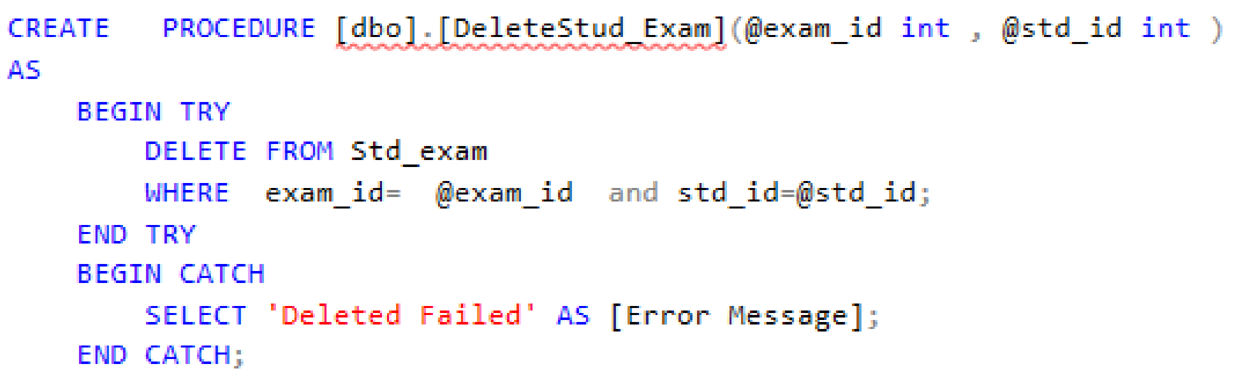
Procedure

dbo.DeleteStud_Exam

Parameter	Data Type	Default	Is Output
@exam_id	int		
@std_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(14)	



```sql
CREATE   PROCEDURE [dbo].[DeleteStud_Exam](@exam_id int , @std_id int )
AS
    BEGIN TRY
        DELETE FROM Std_exam
        WHERE  exam_id=  @exam_id  and std_id=@std_id;
    END TRY
    BEGIN CATCH
        SELECT 'Deleted Failed' AS [Error Message];
    END CATCH;
```

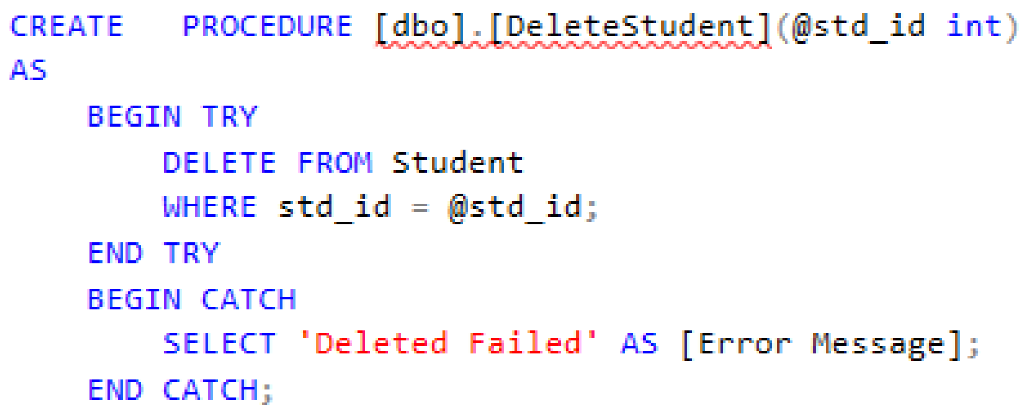
Procedure

dbo.DeleteStudent

Parameter	Data Type	Default	Is Output
@std_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(14)	



```
CREATE   PROCEDURE [dbo].[DeleteStudent](@std_id int)
AS
    BEGIN TRY
        DELETE FROM Student
        WHERE std_id = @std_id;
    END TRY
    BEGIN CATCH
        SELECT 'Deleted Failed' AS [Error Message];
    END CATCH;
```

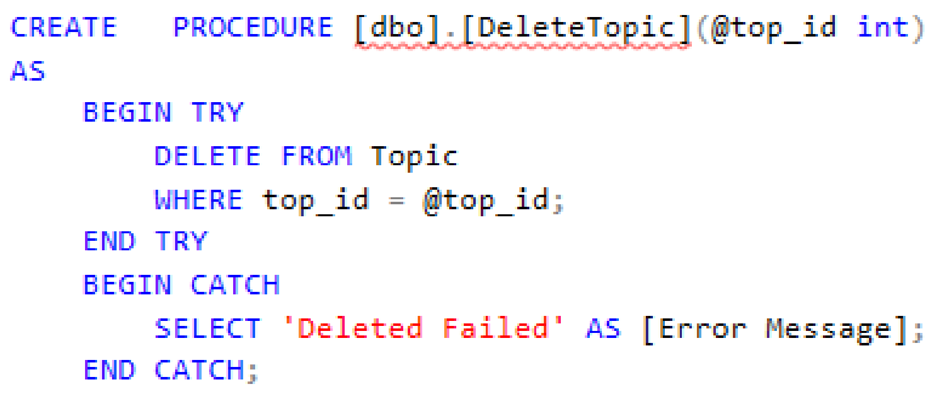
Procedure

dbo.DeleteTopic

Parameter	Data Type	Default	Is Output
@top_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(14)	



```
CREATE   PROCEDURE [dbo].[DeleteTopic](@top_id int)
AS
    BEGIN TRY
        DELETE FROM Topic
        WHERE top_id = @top_id;
    END TRY
    BEGIN CATCH
        SELECT 'Deleted Failed' AS [Error Message];
    END CATCH;
```

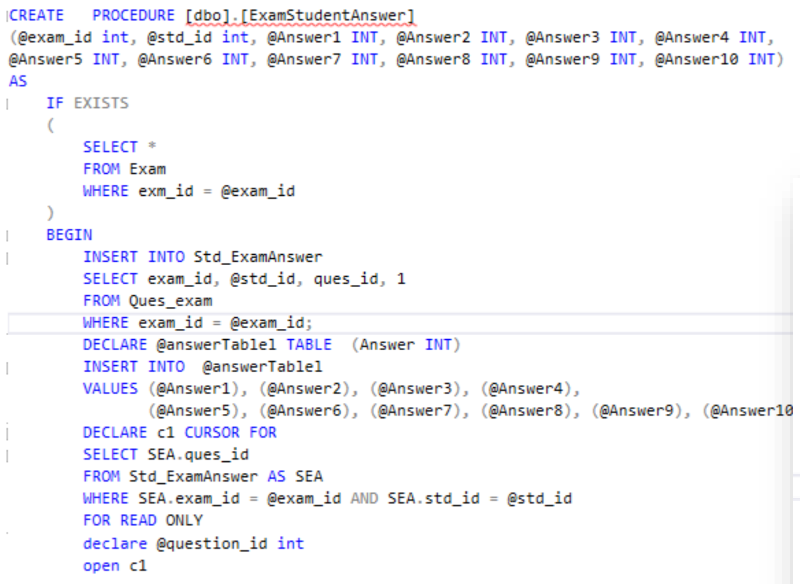
Procedure

dbo.ExamStudentAnswer

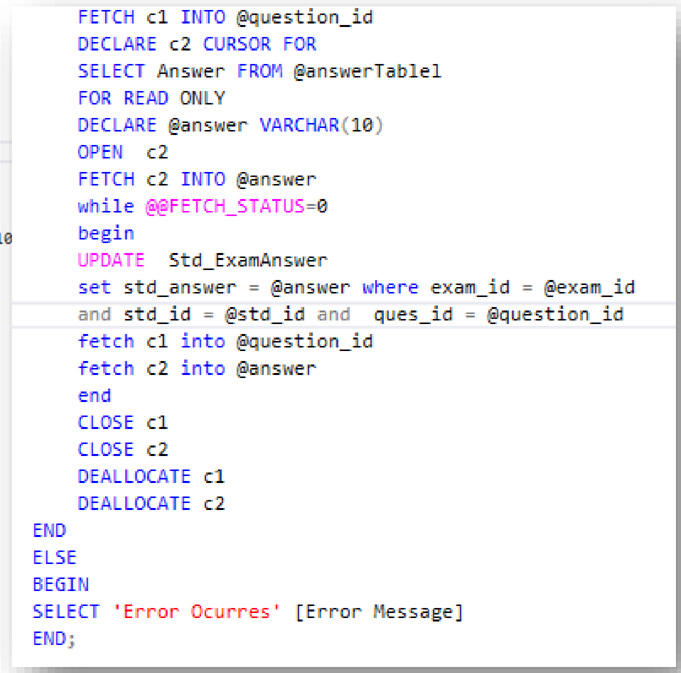
Parameter	Data Type	Default	Is Output
@exam_id	int		
@std_id	int		
@Answer1	int		
@Answer2	int		
@Answer3	int		
@Answer4	int		
@Answer5	int		
@Answer6	int		
@Answer7	int		
@Answer8	int		
@Answer9	int		
@Answer10	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```sql
CREATE   PROCEDURE [dbo].[ExamStudentAnswer]
(@exam_id int, @std_id int, @Answer1 INT, @Answer2 INT, @Answer3 INT, @Answer4 INT,
@Answer5 INT, @Answer6 INT, @Answer7 INT, @Answer8 INT, @Answer9 INT, @Answer10 INT)
AS
    IF EXISTS
    (
        SELECT *
        FROM Exam
        WHERE exm_id = @exam_id
    )
    BEGIN
        INSERT INTO Std_ExamAnswer
        SELECT exam_id, @std_id, ques_id, 1
        FROM Ques_exam
        WHERE exam_id = @exam_id;
        DECLARE @answerTablel TABLE  (Answer INT)
        INSERT INTO  @answerTablel
        VALUES (@Answer1), (@Answer2), (@Answer3), (@Answer4),
               (@Answer5), (@Answer6), (@Answer7), (@Answer8), (@Answer9), (@Answer10
        DECLARE c1 CURSOR FOR
        SELECT SEA.ques_id
        FROM Std_ExamAnswer AS SEA
        WHERE SEA.exam_id = @exam_id AND SEA.std_id = @std_id
        FOR READ ONLY
        declare @question_id int
        open c1
```



```sql
     FETCH c1 INTO @question_id
     DECLARE c2 CURSOR FOR
     SELECT Answer FROM @answerTablel
     FOR READ ONLY
     DECLARE @answer VARCHAR(10)
     OPEN  c2
     FETCH c2 INTO @answer
     while @@FETCH_STATUS=0
     begin
     UPDATE  Std_ExamAnswer
     set std_answer = @answer where exam_id = @exam_id
     and std_id = @std_id and  ques_id = @question_id
     fetch c1 into @question_id
     fetch c2 into @answer
     end
     CLOSE c1
     CLOSE c2
     DEALLOCATE c1
     DEALLOCATE c2
END
ELSE
BEGIN
SELECT 'Error Ocurres' [Error Message]
END;
```

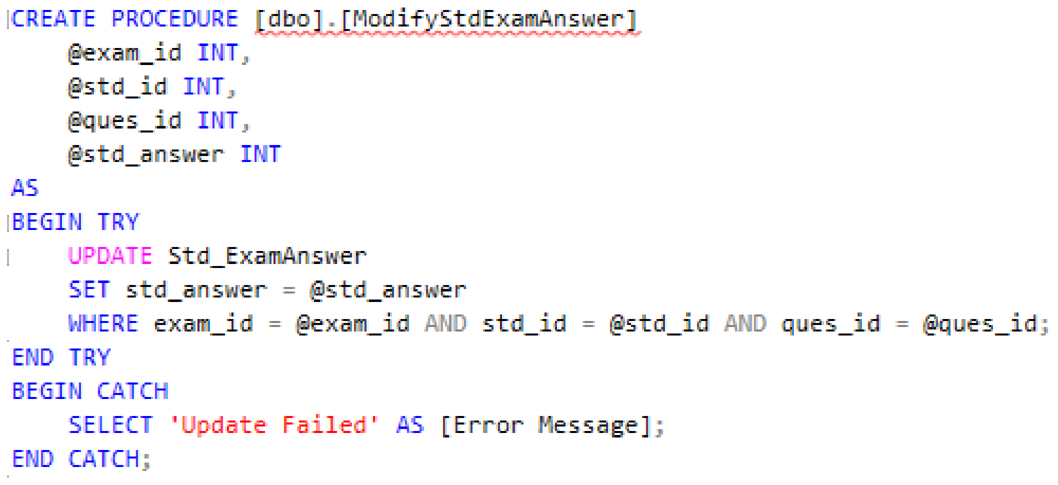
Procedure

dbo.ModifyStdExamAnswer

Parameter	Data Type	Default	Is Output
@exam_id	int		
@std_id	int		
@ques_id	int		
@std_answer	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```sql
CREATE PROCEDURE [dbo].[ModifyStdExamAnswer]
    @exam_id INT,
    @std_id INT,
    @ques_id INT,
    @std_answer INT
AS
BEGIN TRY
    UPDATE Std_ExamAnswer
    SET std_answer = @std_answer
    WHERE exam_id = @exam_id AND std_id = @std_id AND ques_id = @ques_id;
END TRY
BEGIN CATCH
    SELECT 'Update Failed' AS [Error Message];
END CATCH;
```

Procedure

dbo.Optoin2Ques

Parameter	Data Type	Default	Is Output
@q_id	int		
@opt1	nvarchar(10)		
@opt2	nvarchar(10)		
@opt3	nvarchar(30)	Null	
@opt4	nvarchar(30)	Null	

Result:

Column	Data Type	Nullable
Error Message	varchar(26)	

```
create   proc [dbo].[Optoin2Ques]
(@q_id int,@opt1 nvarchar(10), @opt2 nvarchar(10),@opt3 nvarchar(30)=Null,@opt4 nvarchar(30)=Null)
as
    begin try
        if not exists (select 1 from QUESTION where ques_id=@q_id)
             begin
                 select 'Question ID does not exist' AS [Error Message];
                 return;
              end
        if exists (select 1 from Question_Type where ques_id = @q_id)
           begin
                 select 'Options for this Question Already Exsit' AS [Error Message];
                 return;
            end


            insert into Question_Type
            values(@q_id,@opt1,@opt2,@opt3,@opt4)
            select 'Operation completed successfully' AS [Message];
    end try
    begin catch
        select 'Opertion Failed' as [Error Message]
    end catch
```

Procedure

dbo.QuesInEx

Parameter	Data Type	Default	Is Output
@exm_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(30)	

```
create   proc [dbo].[QuesInEx] (@exm_id int)
as
    begin try
        if not exists (select 1 from Ques_exam where exam_id=@exm_id)
                begin
                    select 'There is  no Exam With this ID' as [Error Message]
                    return;
                end


        select  qex.ques_id as Questions , q.Content
        from Ques_exam qex inner join QUESTION q
            on qex.exam_id = @exm_id and qex.ques_id =q.Ques_Id
        select 'Selection Operation completed successfully' AS [Message];
    end try
    begin catch
        select 'Selection Failed' as [Error Message]
    end catch
```

Procedure

bo.ReportExamAndStudentAnswer

Parameter	Data Type	Default	Is Output
@exm_id	int		
@std_id	int		

Result:

Column	Data Type	Nullable
Content	nvarchar(max)	
Student Answer	nvarchar(100)	X

```sql
CREATE   PROCEDURE [dbo].[ReportExamAndStudentAnswer](@exm_id INT, @std_id INT)
AS
    BEGIN TRY
        SELECT Q.Content,
               Options.Answer AS [Student Answer]
        FROM Exam AS E
        INNER JOIN Std_ExamAnswer AS SEA
        ON E.exm_id = SEA.exam_id
        AND SEA.exam_id = @exm_id AND SEA.std_id = @std_id
        INNER JOIN QUESTION AS Q
        ON Q.Ques_Id = SEA.ques_id
        INNER JOIN GetExamOption(@exm_id, @std_id) AS Options
        ON Options.ques_id = SEA.Ques_Id;
        -- Date For Visual Report
        DELETE
        FROM HelperReport
        WHERE ID = 6;
        --
        INSERT INTO HelperReport
            SELECT 6, std_fname + ' ' + std_lname AS [Full Name]
            FROM Student
            WHERE std_id = @std_id;

        INSERT INTO HelperReport
            SELECT 6, crs_name AS [Course Name]
            FROM Exam AS E
            INNER JOIN Course AS C
            ON E.crs_id = C.crs_id
            WHERE E.exm_id = @exm_id;
    END TRY

    BEGIN CATCH
        SELECT 'Error Occured!' AS MessageError;
    END CATCH
```

Procedure

dbo.ReportExamQuestionsChoicesByExamID

Parameter	Data Type	Default	Is Output
@exm_id	int		

Result:

Column	Data Type	Nullable
Ques_Id	int	
Content	nvarchar(max)	
Answer	nvarchar(max)	
Type	varchar(20)	
Opt1	nvarchar(100)	
Opt2	nvarchar(100)	
Opt3	nvarchar(100)	X
Opt4	nvarchar(100)	X

```sql
CREATE   PROCEDURE [dbo].[ReportExamQuestionsChoicesByExamID](@exm_id INT)
AS
    BEGIN TRY
        SELECT Q.*,
               QT.Opt1,
               QT.Opt2,
               QT.Opt3,
               QT.Opt4
        FROM Exam AS E
        INNER JOIN Ques_exam AS QE
        ON E.exm_id = QE.exam_id
        INNER JOIN QUESTION AS Q
        ON Q.Ques_Id = QE.ques_id
        INNER JOIN Question_Type AS QT
        ON Q.Ques_Id =QT.Ques_Id
        WHERE E.exm_id = @exm_id;

        -- Date For Visual Report
        DELETE
        FROM HelperReport
        WHERE ID = 5;

        INSERT INTO HelperReport
            SELECT 5, crs_name AS [Course Name]
            FROM Exam AS E
            INNER JOIN Course AS C
            ON E.crs_id = C.crs_id
            WHERE E.exm_id = @exm_id;

        INSERT INTO HelperReport
            SELECT 5, exm_model AS [Exam Model]
            FROM Exam AS E
            INNER JOIN Course AS C
            ON E.crs_id = C.crs_id
            WHERE E.exm_id = @exm_id;
    END TRY
    BEGIN CATCH
        SELECT 'Error with exm_id' AS MessageError;
    END CATCH
```

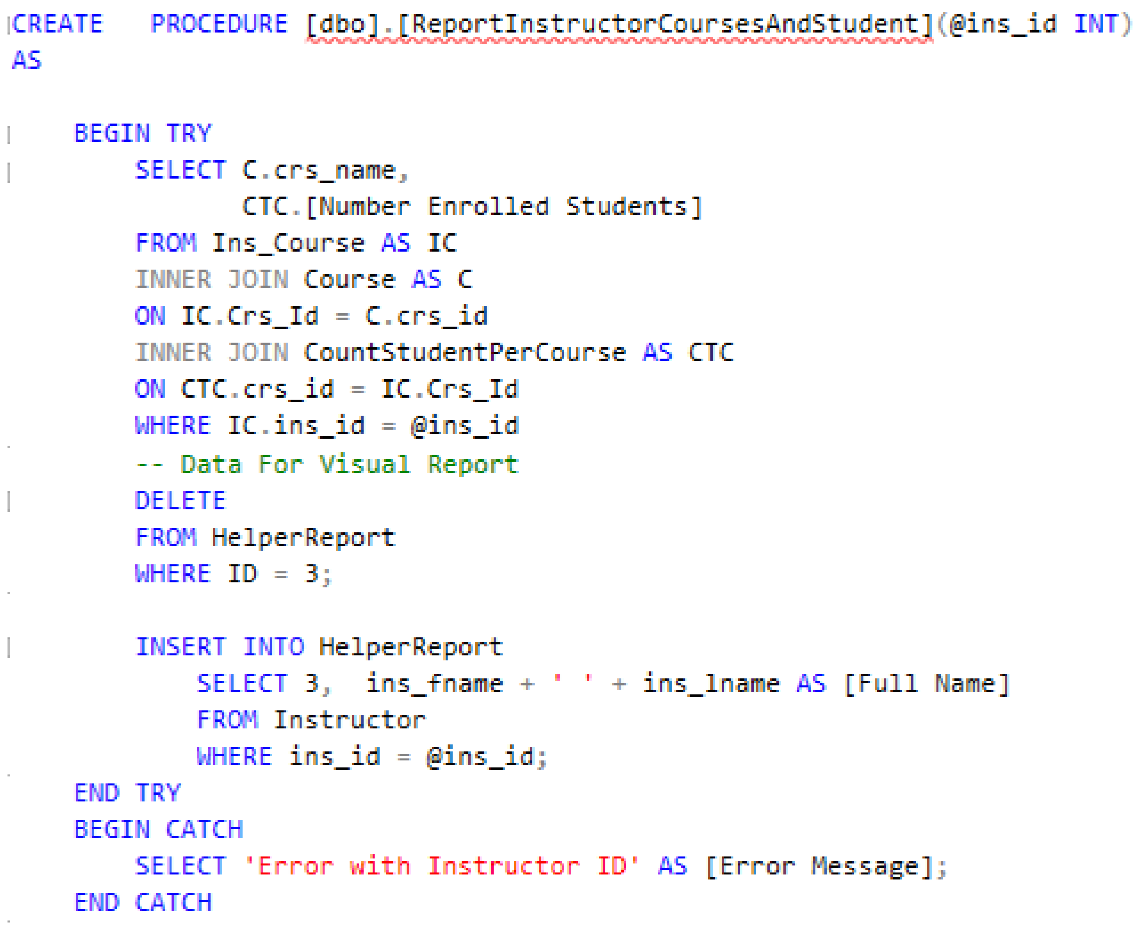
Procedure

dbo.ReportInstructorCoursesAndStudent

Parameter	Data Type	Default	Is Output
@ins_id	int		

Result:

Column	Data Type	Nullable
crs_name	nvarchar(50)	
Number Enrolled Students	int	X



```sql
CREATE   PROCEDURE [dbo].[ReportInstructorCoursesAndStudent](@ins_id INT)
AS

    BEGIN TRY
        SELECT C.crs_name,
               CTC.[Number Enrolled Students]
        FROM Ins_Course AS IC
        INNER JOIN Course AS C
        ON IC.Crs_Id = C.crs_id
        INNER JOIN CountStudentPerCourse AS CTC
        ON CTC.crs_id = IC.Crs_Id
        WHERE IC.ins_id = @ins_id
        -- Data For Visual Report
        DELETE
        FROM HelperReport
        WHERE ID = 3;

        INSERT INTO HelperReport
            SELECT 3,  ins_fname + ' ' + ins_lname AS [Full Name]
            FROM Instructor
            WHERE ins_id = @ins_id;
    END TRY
    BEGIN CATCH
        SELECT 'Error with Instructor ID' AS [Error Message];
    END CATCH
```

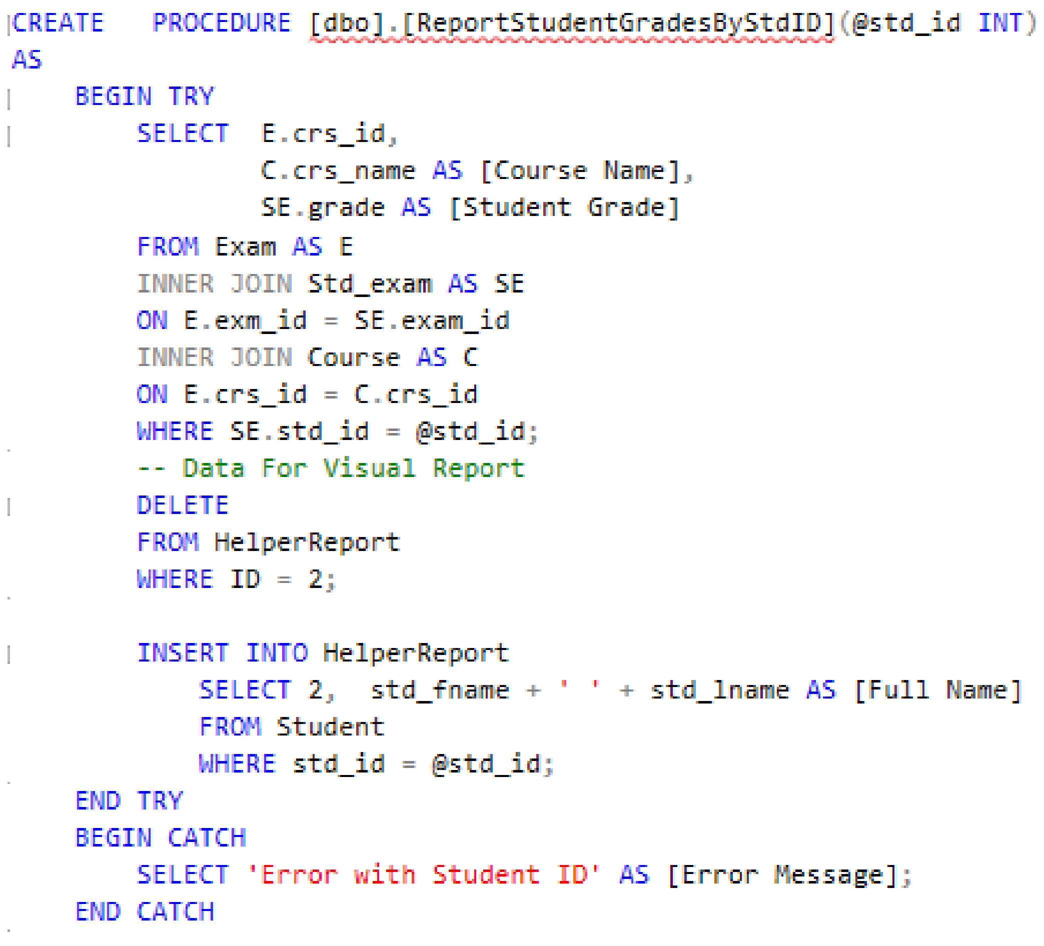
Procedure

dbo.ReportStudentGradesByStdID

Parameter	Data Type	Default	Is Output
@std_id	int		

Result:

Column	Data Type	Nullable
crs_id	int	X
Course Name	nvarchar(50)	
Student Grade	int	



```sql
CREATE   PROCEDURE [dbo].[ReportStudentGradesByStdID](@std_id INT)
AS
    BEGIN TRY
        SELECT  E.crs_id,
                C.crs_name AS [Course Name],
                SE.grade AS [Student Grade]
        FROM Exam AS E
        INNER JOIN Std_exam AS SE
        ON E.exm_id = SE.exam_id
        INNER JOIN Course AS C
        ON E.crs_id = C.crs_id
        WHERE SE.std_id = @std_id;
        -- Data For Visual Report
        DELETE
        FROM HelperReport
        WHERE ID = 2;

        INSERT INTO HelperReport
            SELECT 2,  std_fname + ' ' + std_lname AS [Full Name]
            FROM Student
            WHERE std_id = @std_id;
    END TRY
    BEGIN CATCH
        SELECT 'Error with Student ID' AS [Error Message];
    END CATCH
```

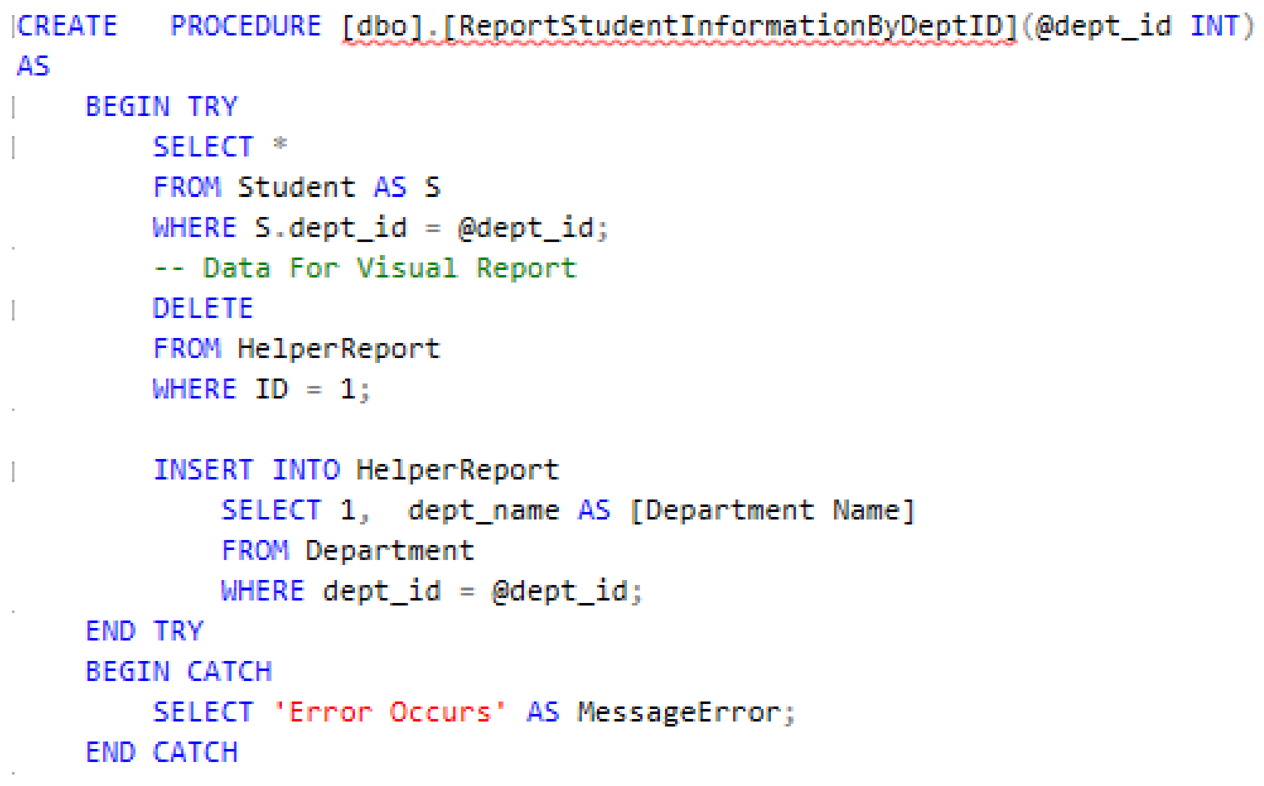
Procedure

dbo.ReportStudentInformationByDeptID

Parameter	Data Type	Default	Is Output
@dept_id	int		

Result:

Column	Data Type	Nullable
std_id	int	
std_fname	nvarchar(20)	
std_lname	nvarchar(20)	
city	nvarchar(20)	
street	nvarchar(50)	X
phone	varchar(11)	
std_DoB	date	
email	nvarchar(50)	
passowrd	nvarchar(20)	
dept_id	int	X
gender	nvarchar(3)	



```
CREATE   PROCEDURE [dbo].[ReportStudentInformationByDeptID](@dept_id INT)
AS
    BEGIN TRY
        SELECT *
        FROM Student AS S
        WHERE S.dept_id = @dept_id;
        -- Data For Visual Report
        DELETE
        FROM HelperReport
        WHERE ID = 1;

        INSERT INTO HelperReport
            SELECT 1,  dept_name AS [Department Name]
            FROM Department
            WHERE dept_id = @dept_id;
    END TRY
    BEGIN CATCH
        SELECT 'Error Occurs' AS MessageError;
    END CATCH
```

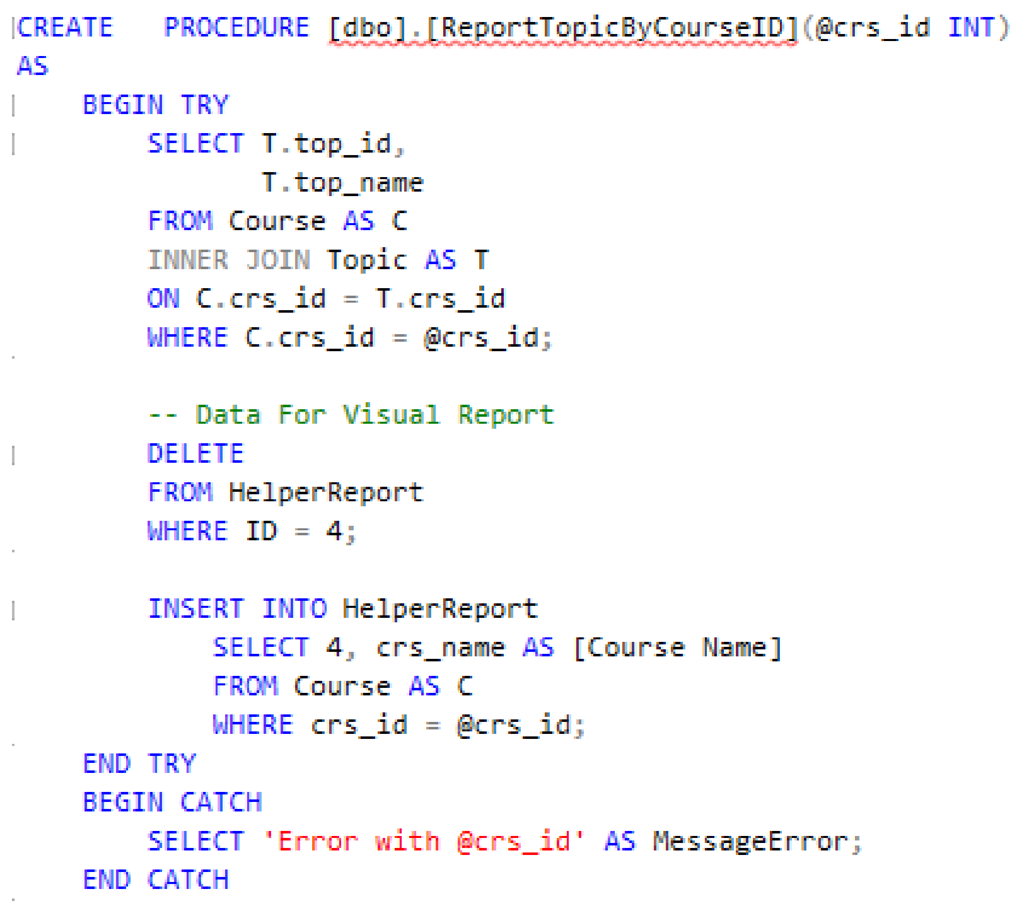
Procedure

dbo.ReportTopicByCourseID

Parameter	Data Type	Default	Is Output
@crs_id	int		

Result:

Column	Data Type	Nullable
top_id	int	
top_name	nvarchar(50)	



```
CREATE   PROCEDURE [dbo].[ReportTopicByCourseID](@crs_id INT)
AS
    BEGIN TRY
        SELECT T.top_id,
               T.top_name
        FROM Course AS C
        INNER JOIN Topic AS T
        ON C.crs_id = T.crs_id
        WHERE C.crs_id = @crs_id;

        -- Data For Visual Report
        DELETE
        FROM HelperReport
        WHERE ID = 4;

        INSERT INTO HelperReport
            SELECT 4, crs_name AS [Course Name]
            FROM Course AS C
            WHERE crs_id = @crs_id;
    END TRY
    BEGIN CATCH
        SELECT 'Error with @crs_id' AS MessageError;
    END CATCH
```

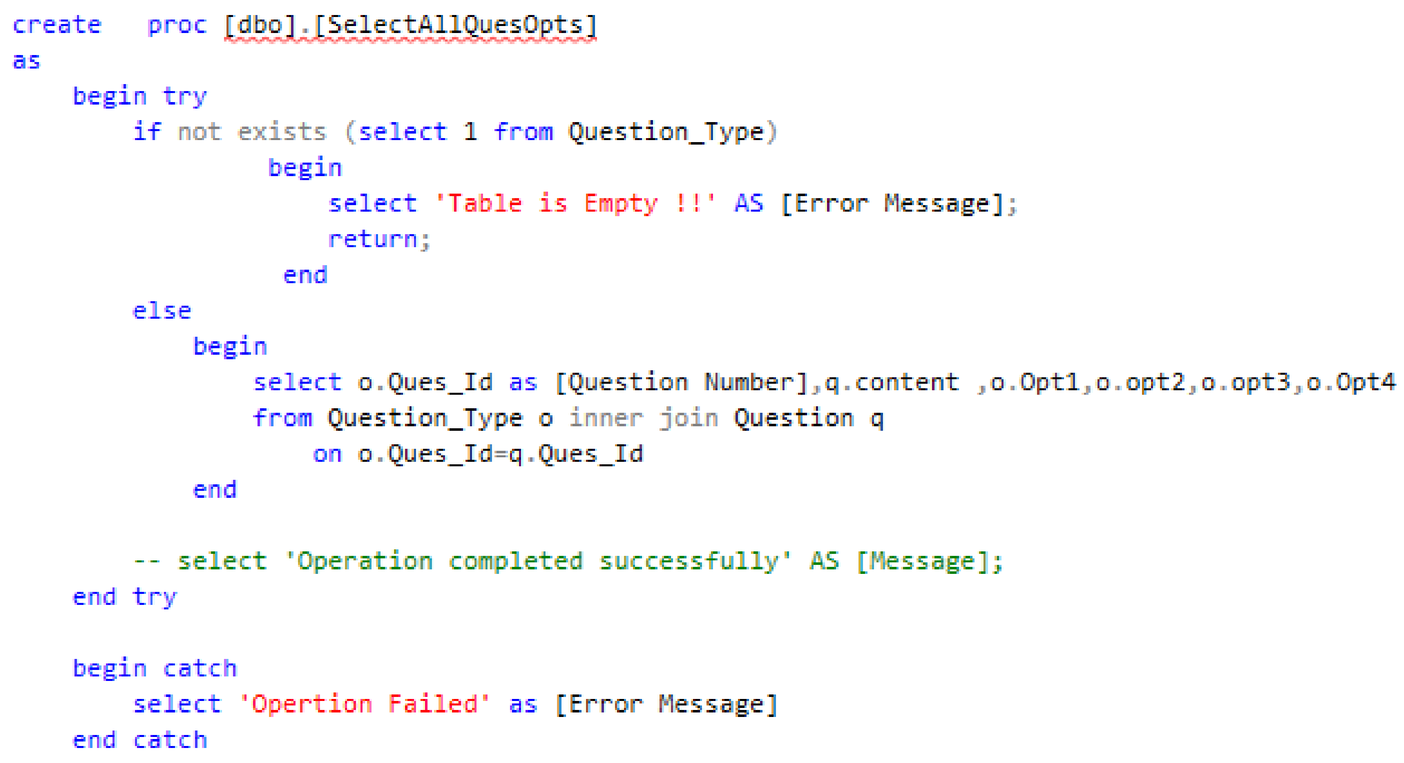
Procedure

dbo.SelectAllQuesOpts

No parameters.

Result:

Column	Data Type	Nullable
Error Message	varchar(17)	



```
create   proc [dbo].[SelectAllQuesOpts]
as
    begin try
        if not exists (select 1 from Question_Type)
                 begin
                     select 'Table is Empty !!' AS [Error Message];
                     return;
                  end
        else
            begin
                select o.Ques_Id as [Question Number],q.content ,o.Opt1,o.opt2,o.opt3,o.Opt4
                from Question_Type o inner join Question q
                    on o.Ques_Id=q.Ques_Id
            end

        -- select 'Operation completed successfully' AS [Message];
    end try

    begin catch
        select 'Opertion Failed' as [Error Message]
    end catch
```

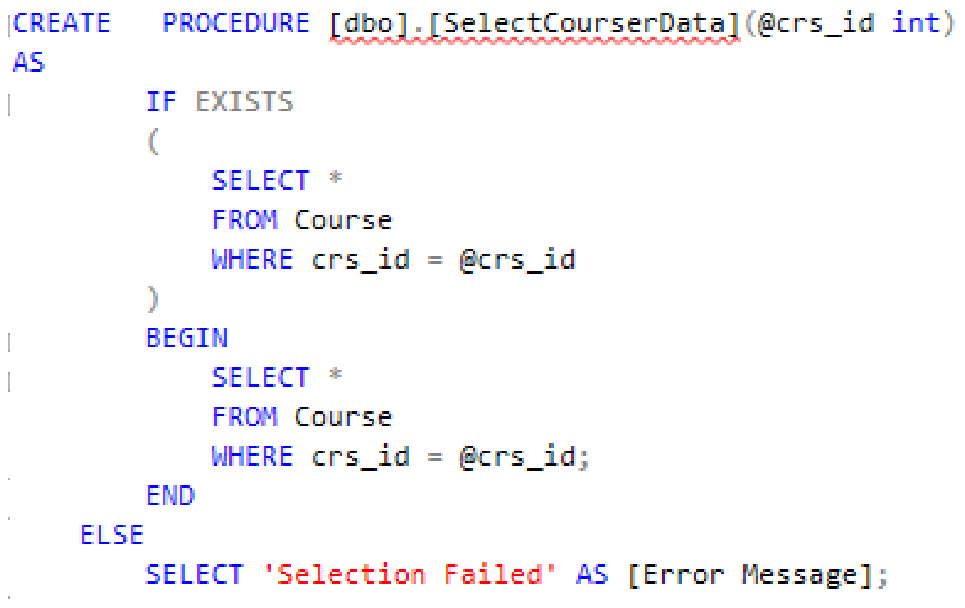
Procedure

dbo.SelectCourserData

Parameter	Data Type	Default	Is Output
@crs_id	int		

Result:

Column	Data Type	Nullable
crs_id	int	
crs_name	nvarchar(50)	
hours	int	



```sql
CREATE   PROCEDURE [dbo].[SelectCourserData](@crs_id int)
AS
        IF EXISTS
        (
            SELECT *
            FROM Course
            WHERE crs_id = @crs_id
        )
        BEGIN
            SELECT *
            FROM Course
            WHERE crs_id = @crs_id;
        END
    ELSE
        SELECT 'Selection Failed' AS [Error Message];
```

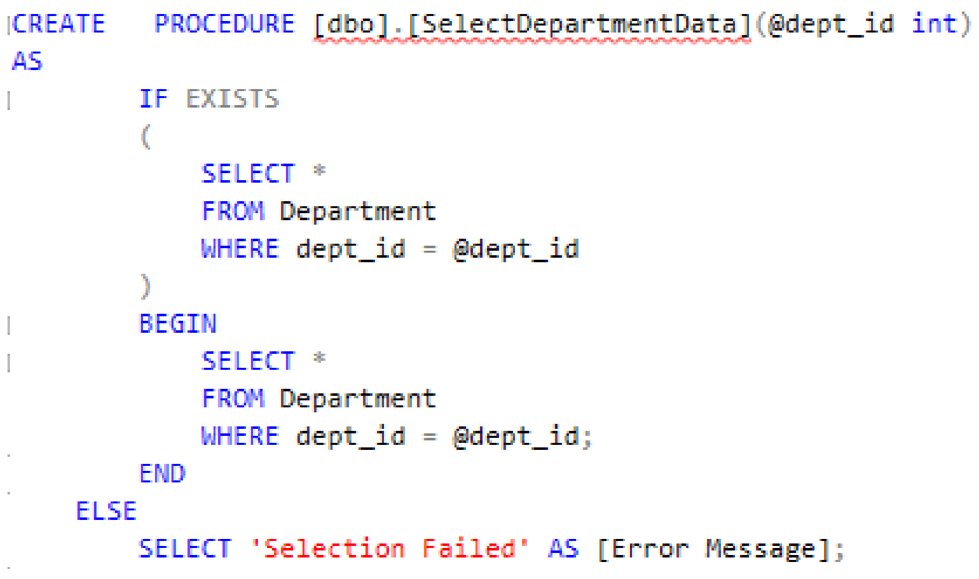
Procedure

dbo.SelectDepartmentData

Parameter	Data Type	Default	Is Output
@dept_id	int		

Result:

Column	Data Type	Nullable
dept_id	int	
dept_name	nvarchar(50)	
dept_location	nvarchar(50)	
phone	varchar(11)	
mgr_id	int	X
mgr_hiredate	date	X



```
CREATE   PROCEDURE [dbo].[SelectDepartmentData](@dept_id int)
AS
        IF EXISTS
        (
            SELECT *
            FROM Department
            WHERE dept_id = @dept_id
        )
        BEGIN
            SELECT *
            FROM Department
            WHERE dept_id = @dept_id;
        END
    ELSE
        SELECT 'Selection Failed' AS [Error Message];
```

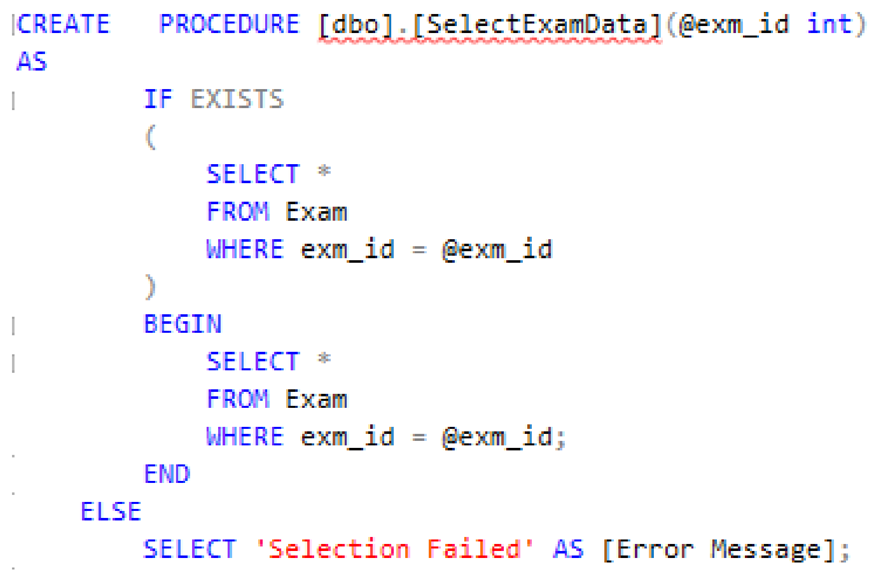
Procedure

dbo.SelectExamData

Parameter	Data Type	Default	Is Output
@exm_id	int		

Result:

Column	Data Type	Nullable
exm_id	int	
exm_model	int	
duration	int	
crs_id	int	X



```
CREATE   PROCEDURE [dbo].[SelectExamData](@exm_id int)
AS
        IF EXISTS
        (
            SELECT *
            FROM Exam
            WHERE exm_id = @exm_id
        )
        BEGIN
            SELECT *
            FROM Exam
            WHERE exm_id = @exm_id;
        END
    ELSE
        SELECT 'Selection Failed' AS [Error Message];
```

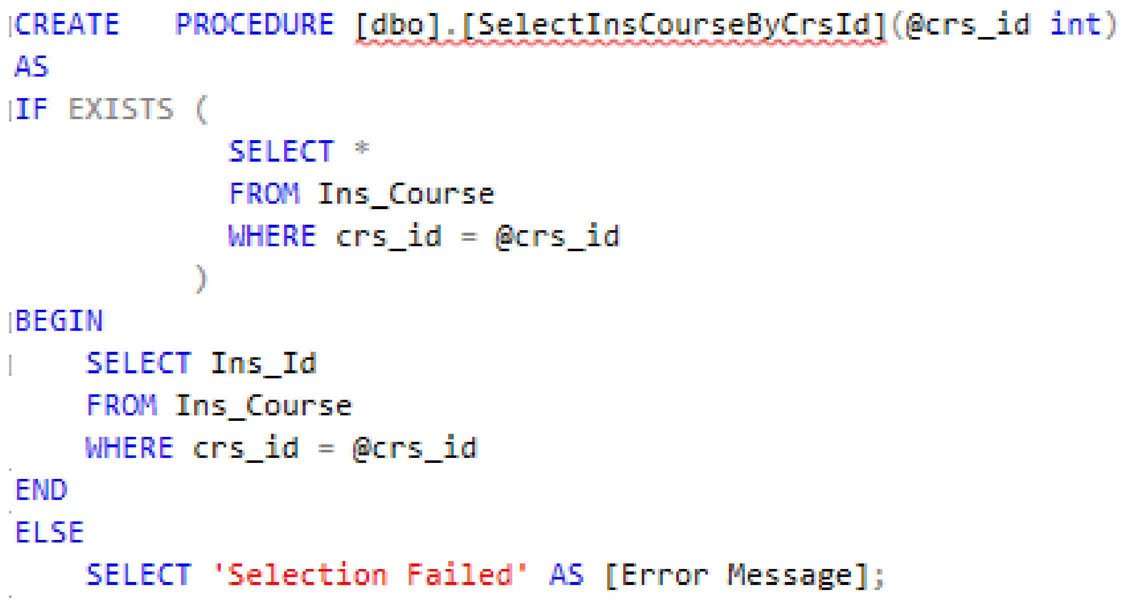
Procedure

dbo.SelectInsCourseByCrsId

Parameter	Data Type	Default	Is Output
@crs_id	int		

Result:

Column	Data Type	Nullable
Ins_Id	int	



```
CREATE   PROCEDURE [dbo].[SelectInsCourseByCrsId](@crs_id int)
AS
IF EXISTS (
            SELECT *
            FROM Ins_Course
            WHERE crs_id = @crs_id
          )
BEGIN
    SELECT Ins_Id
    FROM Ins_Course
    WHERE crs_id = @crs_id
END
ELSE
    SELECT 'Selection Failed' AS [Error Message];
```

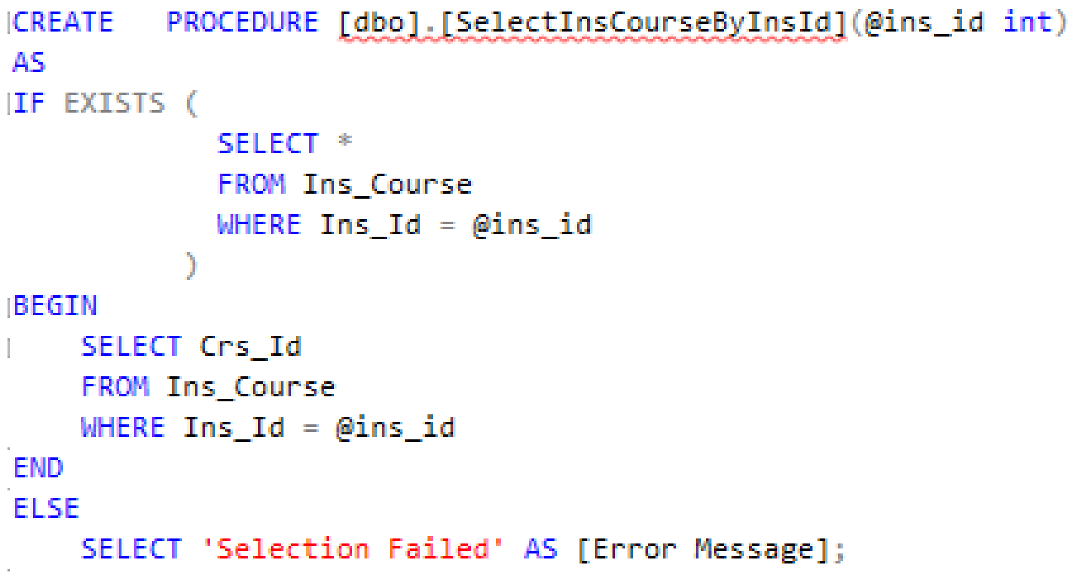
Procedure

dbo.SelectInsCourseByInsId

Parameter	Data Type	Default	Is Output
@ins_id	int		

Result:

Column	Data Type	Nullable
Crs_Id	int	



```sql
CREATE   PROCEDURE [dbo].[SelectInsCourseByInsId](@ins_id int)
AS
IF EXISTS (
            SELECT *
            FROM Ins_Course
            WHERE Ins_Id = @ins_id
          )
BEGIN
    SELECT Crs_Id
    FROM Ins_Course
    WHERE Ins_Id = @ins_id
END
ELSE
    SELECT 'Selection Failed' AS [Error Message];
```

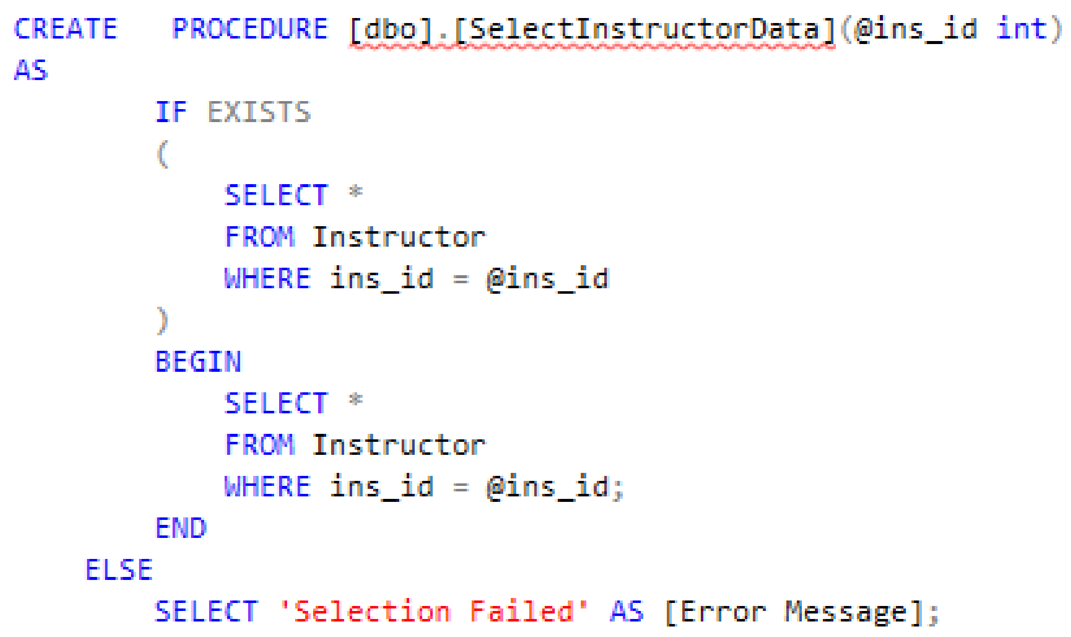
Procedure

dbo.SelectInstructorData

Parameter	Data Type	Default	Is Output
@ins_id	int		

Result:

Column	Data Type	Nullable
ins_id	int	
ins_fname	nvarchar(20)	
ins_lname	nvarchar(20)	
city	nvarchar(20)	
street	nvarchar(20)	X
phone	varchar(11)	
ins_DoB	date	
email	nvarchar(50)	
passowrd	nvarchar(20)	
salary	decimal(10,2)	X
hire_date	date	X
dept_id	int	X
gender	nvarchar(10)	



```
CREATE   PROCEDURE [dbo].[SelectInstructorData](@ins_id int)
AS
        IF EXISTS
        (
            SELECT *
            FROM Instructor
            WHERE ins_id = @ins_id
        )
        BEGIN
            SELECT *
            FROM Instructor
            WHERE ins_id = @ins_id;
        END
    ELSE
        SELECT 'Selection Failed' AS [Error Message];
```

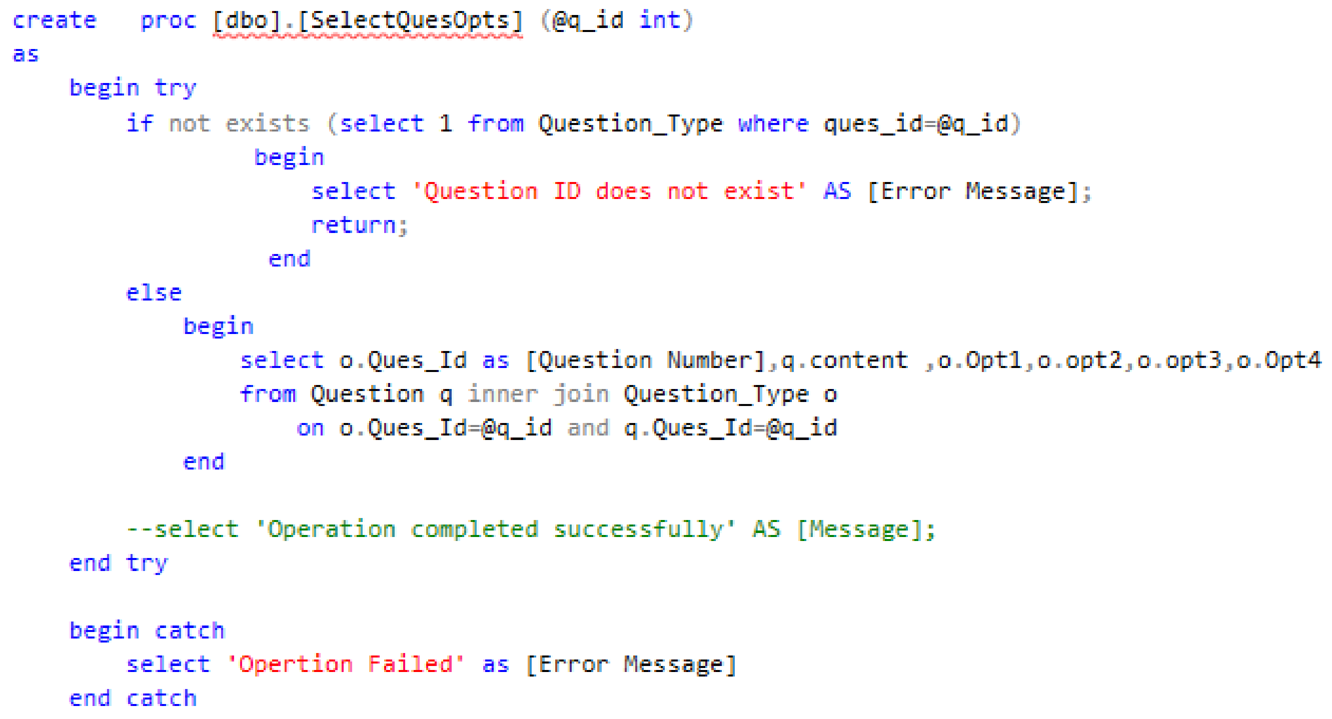
Procedure

dbo.SelectQuesOpts

Parameter	Data Type	Default	Is Output
@q_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(26)	



```
create   proc [dbo].[SelectQuesOpts] (@q_id int)
as
    begin try
        if not exists (select 1 from Question_Type where ques_id=@q_id)
                 begin
                     select 'Question ID does not exist' AS [Error Message];
                     return;
                  end
        else
            begin
                select o.Ques_Id as [Question Number],q.content ,o.Opt1,o.opt2,o.opt3,o.Opt4
                from Question q inner join Question_Type o
                    on o.Ques_Id=@q_id and q.Ques_Id=@q_id
            end

        --select 'Operation completed successfully' AS [Message];
    end try

    begin catch
        select 'Opertion Failed' as [Error Message]
    end catch
```

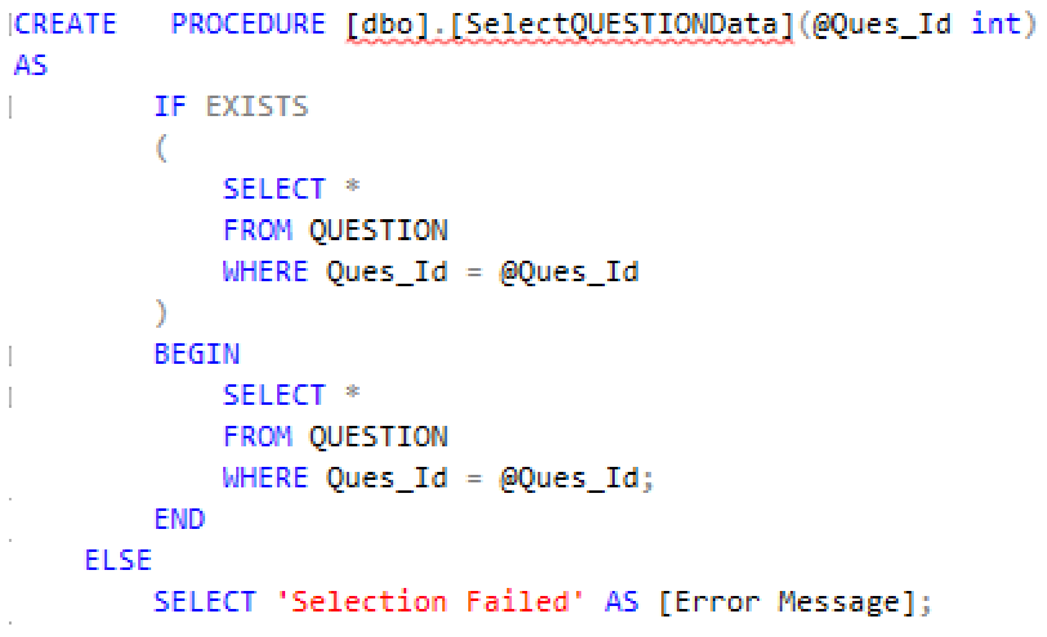
Procedure

dbo.SelectQUESTIONData

Parameter	Data Type	Default	Is Output
@Ques_Id	int		

Result:

Column	Data Type	Nullable
Ques_Id	int	
Content	nvarchar(max)	
Answer	nvarchar(max)	
Type	varchar(20)	



```
CREATE   PROCEDURE [dbo].[SelectQUESTIONData](@Ques_Id int)
AS
        IF EXISTS
        (
            SELECT *
            FROM QUESTION
            WHERE Ques_Id = @Ques_Id
        )
        BEGIN
            SELECT *
            FROM QUESTION
            WHERE Ques_Id = @Ques_Id;
        END
    ELSE
        SELECT 'Selection Failed' AS [Error Message];
```

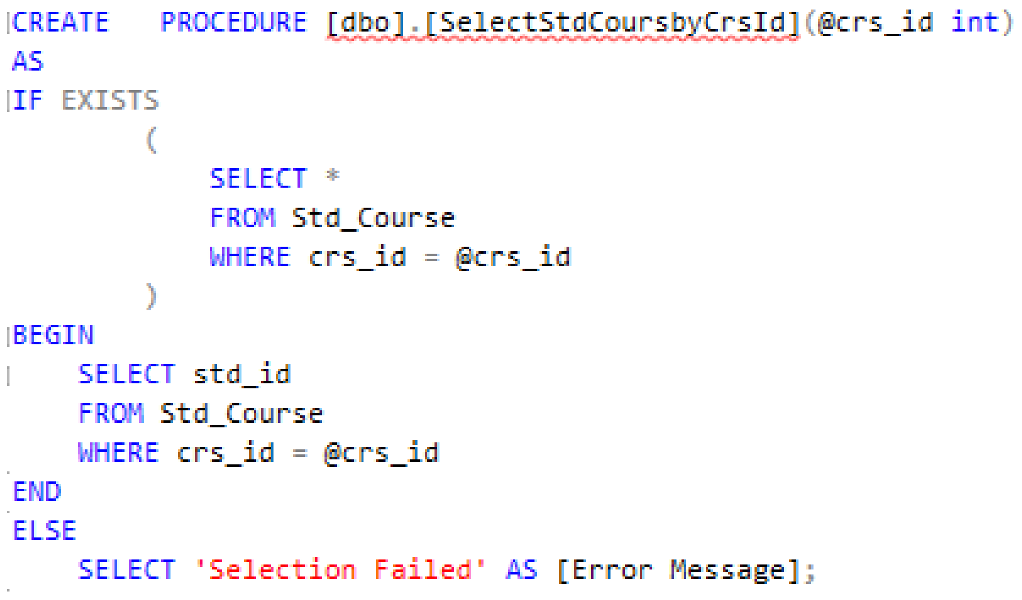
Procedure

dbo.SelectStdCoursbyCrsId

Parameter	Data Type	Default	Is Output
@crs_id	int		

Result:

Column	Data Type	Nullable
std_id	int	



```
CREATE   PROCEDURE [dbo].[SelectStdCoursbyCrsId](@crs_id int)
AS
IF EXISTS
        (
            SELECT *
            FROM Std_Course
            WHERE crs_id = @crs_id
        )
BEGIN
    SELECT std_id
    FROM Std_Course
    WHERE crs_id = @crs_id
END
ELSE
    SELECT 'Selection Failed' AS [Error Message];
```

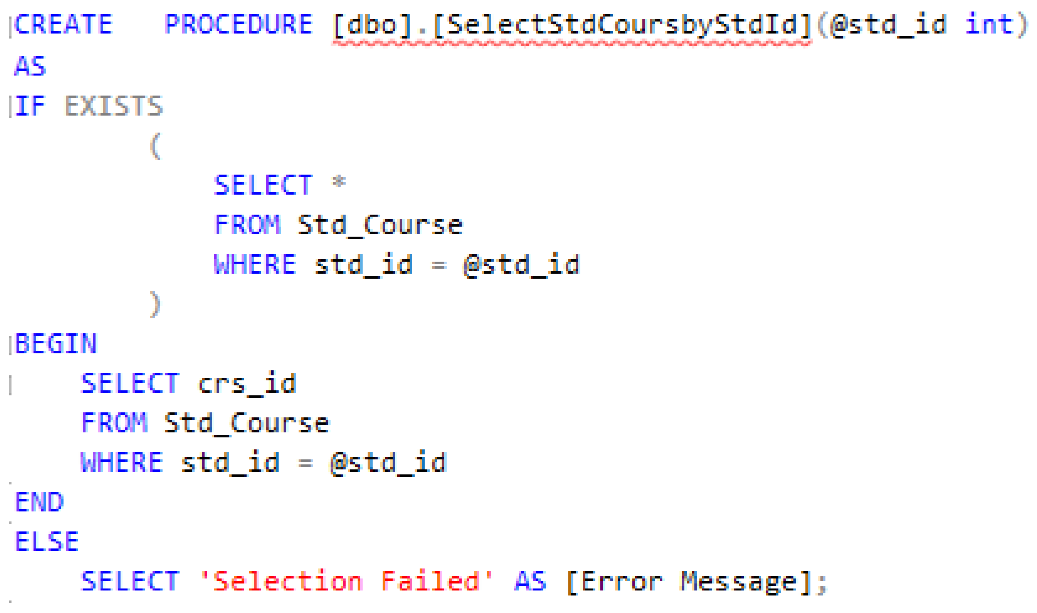
Procedure

dbo.SelectStdCoursbyStdId

Parameter	Data Type	Default	Is Output
@std_id	int		

Result:

Column	Data Type	Nullable
crs_id	int	



```
CREATE   PROCEDURE [dbo].[SelectStdCoursbyStdId](@std_id int)
AS
IF EXISTS
        (
            SELECT *
            FROM Std_Course
            WHERE std_id = @std_id
        )
BEGIN
    SELECT crs_id
    FROM Std_Course
    WHERE std_id = @std_id
END
ELSE
    SELECT 'Selection Failed' AS [Error Message];
```

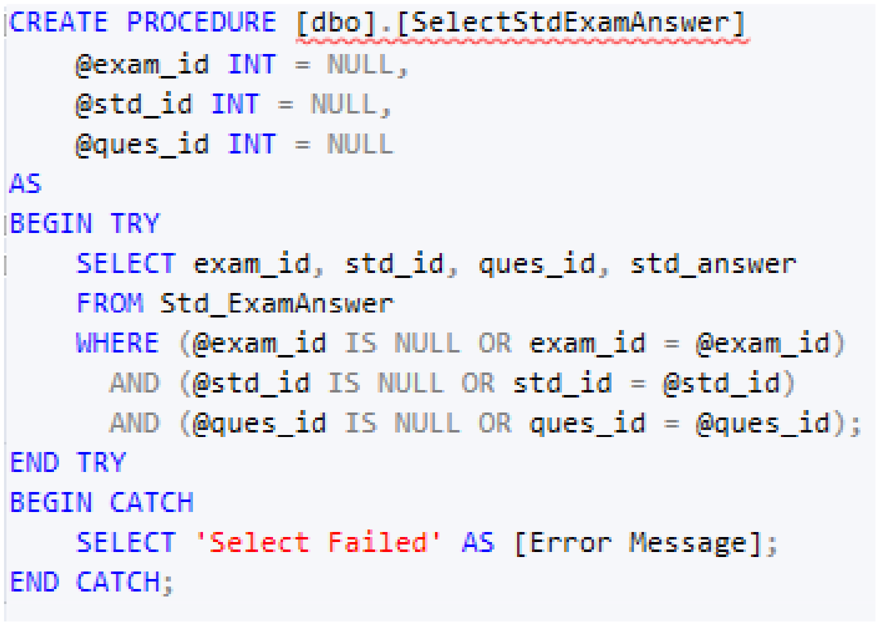
Procedure

dbo.SelectStdExamAnswer

Parameter	Data Type	Default	Is Output
@exam_id	int	NULL	
@std_id	int	NULL	
@ques_id	int	NULL	

Result:

Column	Data Type	Nullable
exam_id	int	
std_id	int	
ques_id	int	
std_answer	int	



```
CREATE PROCEDURE [dbo].[SelectStdExamAnswer]
    @exam_id INT = NULL,
    @std_id INT = NULL,
    @ques_id INT = NULL
AS
BEGIN TRY
    SELECT exam_id, std_id, ques_id, std_answer
    FROM Std_ExamAnswer
    WHERE (@exam_id IS NULL OR exam_id = @exam_id)
      AND (@std_id IS NULL OR std_id = @std_id)
      AND (@ques_id IS NULL OR ques_id = @ques_id);
END TRY
BEGIN CATCH
    SELECT 'Select Failed' AS [Error Message];
END CATCH;
```

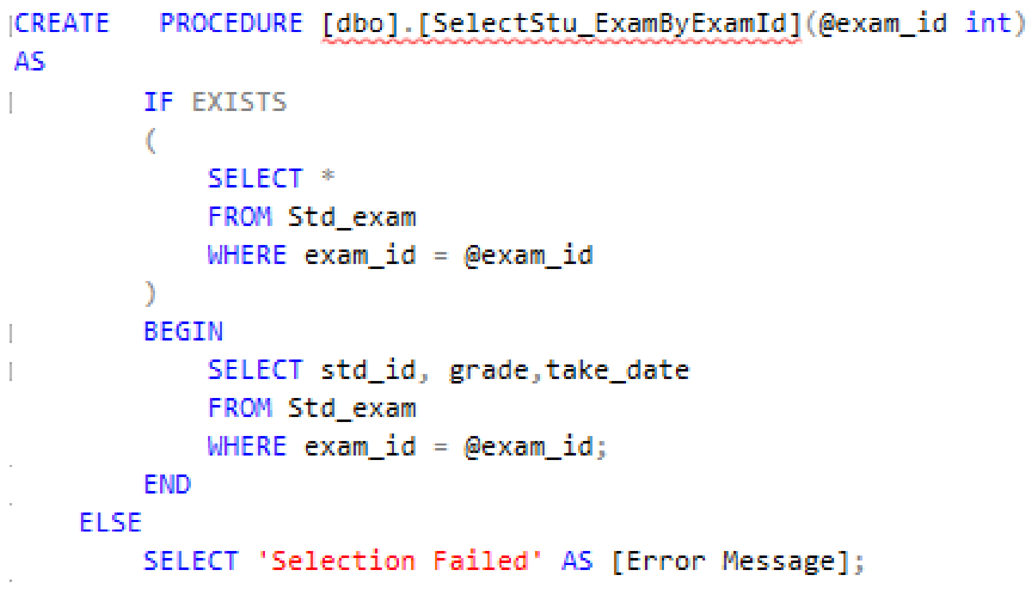
Procedure

dbo.SelectStu_ExamByExamId

Parameter	Data Type	Default	Is Output
@exam_id	int		

Result:

Column	Data Type	Nullable
std_id	int	
grade	int	
take_date	date	



```
CREATE   PROCEDURE [dbo].[SelectStu_ExamByExamId](@exam_id int)
AS
        IF EXISTS
        (
            SELECT *
            FROM Std_exam
            WHERE exam_id = @exam_id
        )
        BEGIN
            SELECT std_id, grade,take_date
            FROM Std_exam
            WHERE exam_id = @exam_id;
        END
    ELSE
        SELECT 'Selection Failed' AS [Error Message];
```

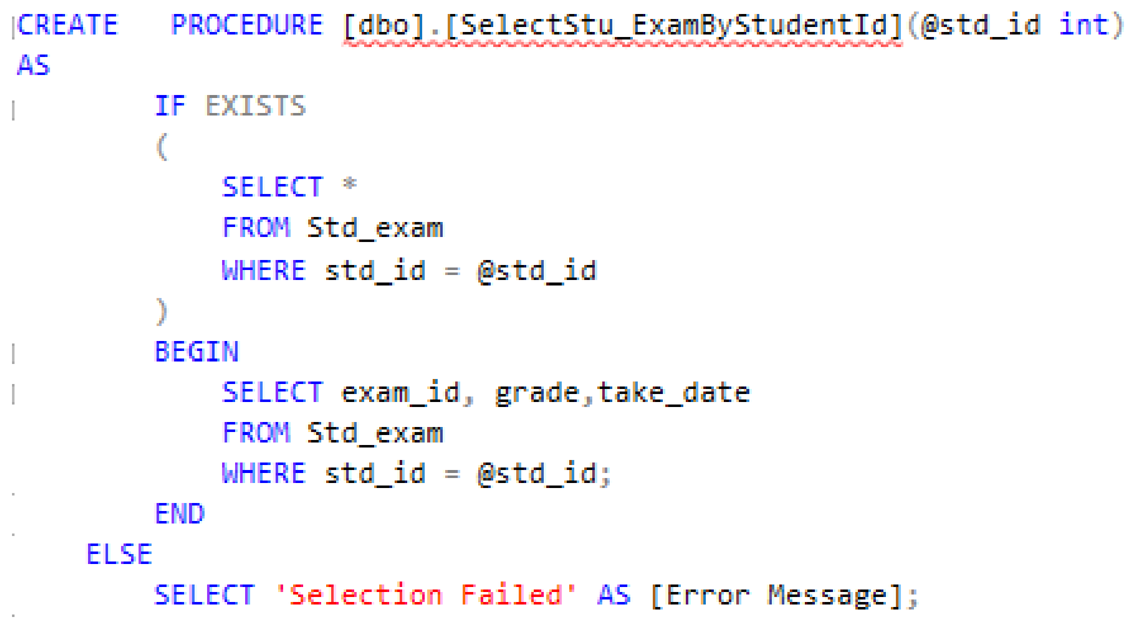
Procedure

dbo.SelectStu_ExamByStudentId

Parameter	Data Type	Default	Is Output
@std_id	int		

Result:

Column	Data Type	Nullable
exam_id	int	
grade	int	
take_date	date	



```
CREATE   PROCEDURE [dbo].[SelectStu_ExamByStudentId](@std_id int)
AS
        IF EXISTS
        (
            SELECT *
            FROM Std_exam
            WHERE std_id = @std_id
        )
        BEGIN
            SELECT exam_id, grade,take_date
            FROM Std_exam
            WHERE std_id = @std_id;
        END
    ELSE
        SELECT 'Selection Failed' AS [Error Message];
```

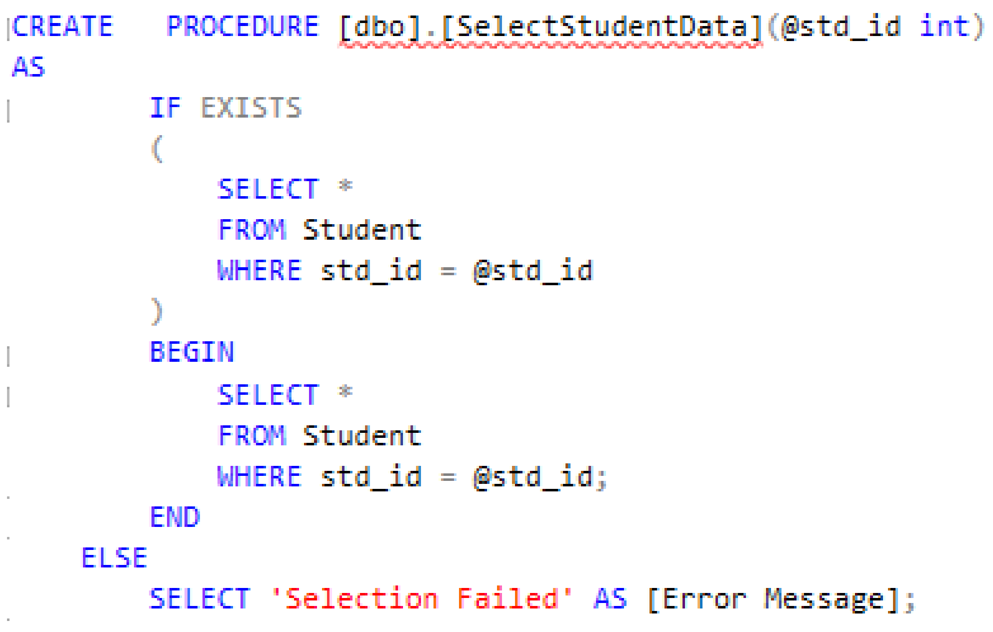
Procedure

dbo.SelectStudentData

Parameter	Data Type	Default	Is Output
@std_id	int		

Result:

Column	Data Type	Nullable
std_id	int	
std_fname	nvarchar(20)	
std_lname	nvarchar(20)	
city	nvarchar(20)	
street	nvarchar(50)	X
phone	varchar(11)	
std_DoB	date	
email	nvarchar(50)	
passowrd	nvarchar(20)	
dept_id	int	X
gender	nvarchar(3)	



```
CREATE   PROCEDURE [dbo].[SelectStudentData](@std_id int)
AS
        IF EXISTS
        (
            SELECT *
            FROM Student
            WHERE std_id = @std_id
        )
        BEGIN
            SELECT *
            FROM Student
            WHERE std_id = @std_id;
        END
    ELSE
        SELECT 'Selection Failed' AS [Error Message];
```

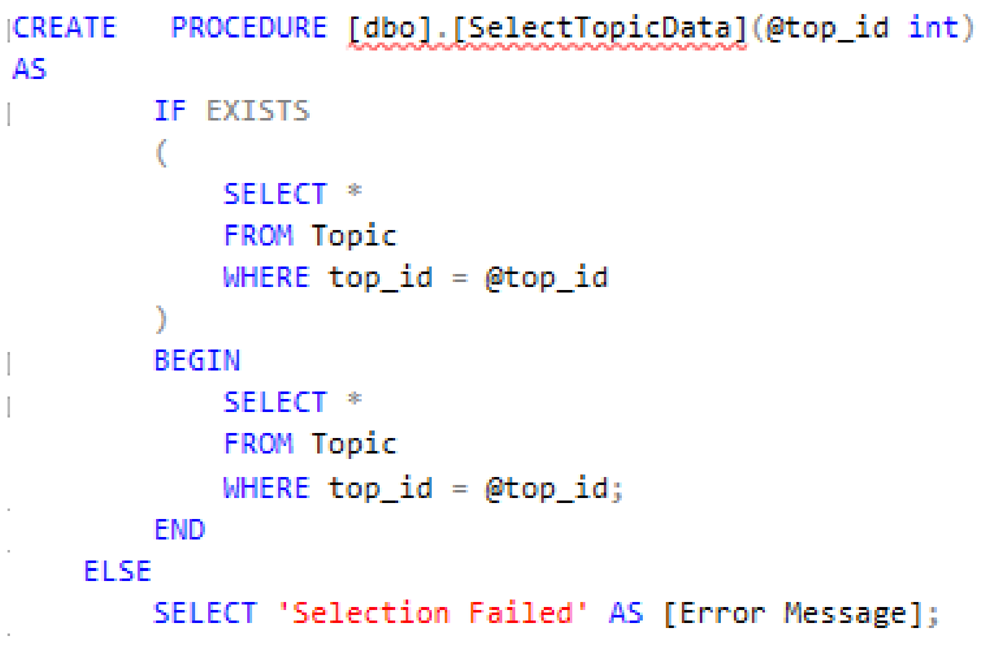
Procedure

dbo.SelectTopicData

Parameter	Data Type	Default	Is Output
@top_id	int		

Result:

Column	Data Type	Nullable
top_id	int	
top_name	nvarchar(50)	
crs_id	int	X



```
CREATE   PROCEDURE [dbo].[SelectTopicData](@top_id int)
AS
        IF EXISTS
        (
            SELECT *
            FROM Topic
            WHERE top_id = @top_id
        )
        BEGIN
            SELECT *
            FROM Topic
            WHERE top_id = @top_id;
        END
    ELSE
        SELECT 'Selection Failed' AS [Error Message];
```

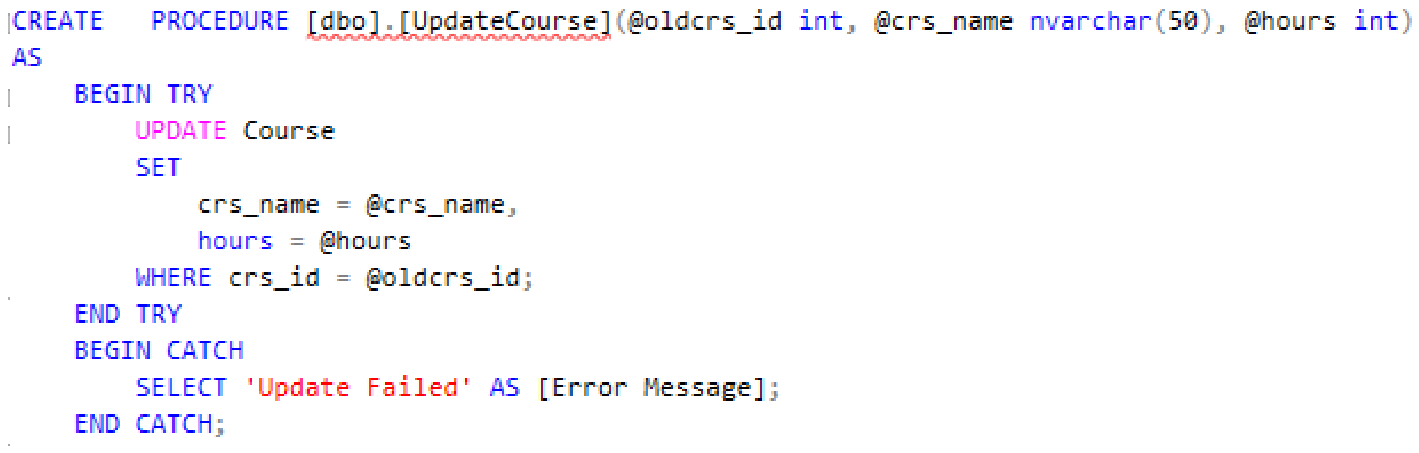
Procedure

dbo.UpdateCourse

Parameter	Data Type	Default	Is Output
@oldcrs_id	int		
@crs_name	nvarchar(50)		
@hours	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```sql
CREATE   PROCEDURE [dbo].[UpdateCourse](@oldcrs_id int, @crs_name nvarchar(50), @hours int)
AS
    BEGIN TRY
        UPDATE Course
        SET
            crs_name = @crs_name,
            hours = @hours
        WHERE crs_id = @oldcrs_id;
    END TRY
    BEGIN CATCH
        SELECT 'Update Failed' AS [Error Message];
    END CATCH;
```

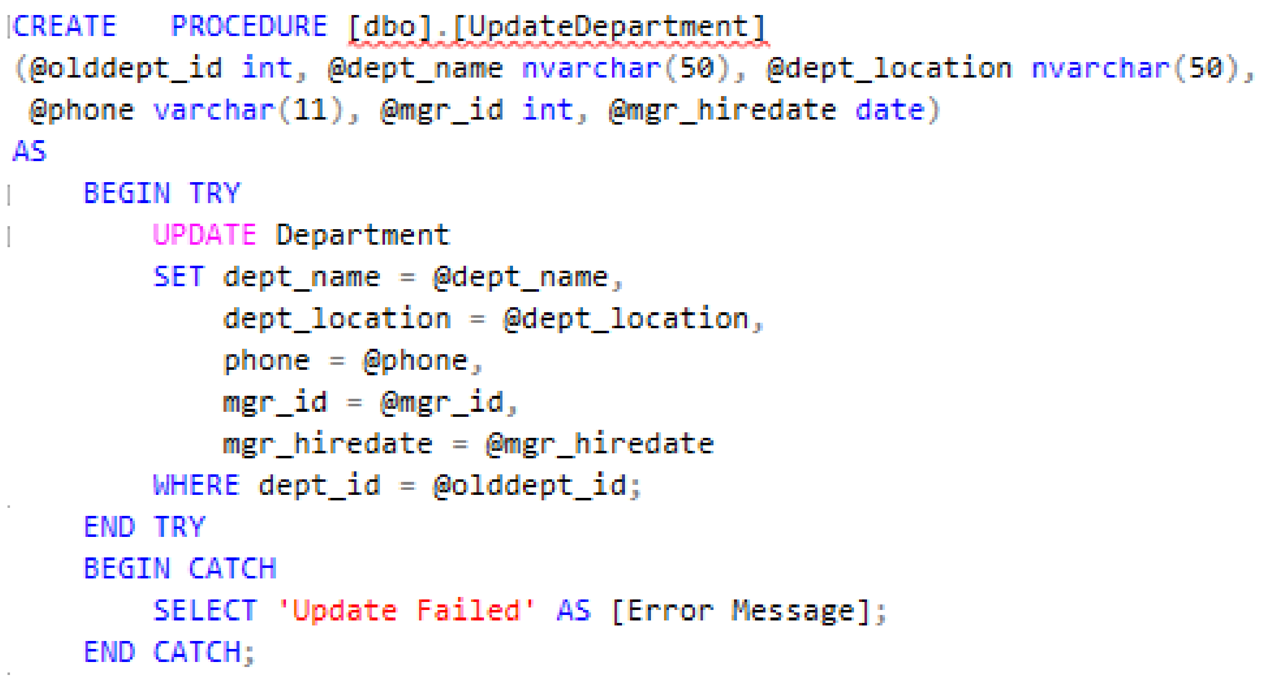
Procedure

dbo.UpdateDepartment

Parameter	Data Type	Default	Is Output
@olddept_id	int		
@dept_name	nvarchar(50)		
@dept_location	nvarchar(50)		
@phone	varchar(11)		
@mgr_id	int		
@mgr_hiredate	date		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```
CREATE   PROCEDURE [dbo].[UpdateDepartment]
(@olddept_id int, @dept_name nvarchar(50), @dept_location nvarchar(50),
 @phone varchar(11), @mgr_id int, @mgr_hiredate date)
AS
    BEGIN TRY
        UPDATE Department
        SET dept_name = @dept_name,
            dept_location = @dept_location,
            phone = @phone,
            mgr_id = @mgr_id,
            mgr_hiredate = @mgr_hiredate
        WHERE dept_id = @olddept_id;
    END TRY
    BEGIN CATCH
        SELECT 'Update Failed' AS [Error Message];
    END CATCH;
```

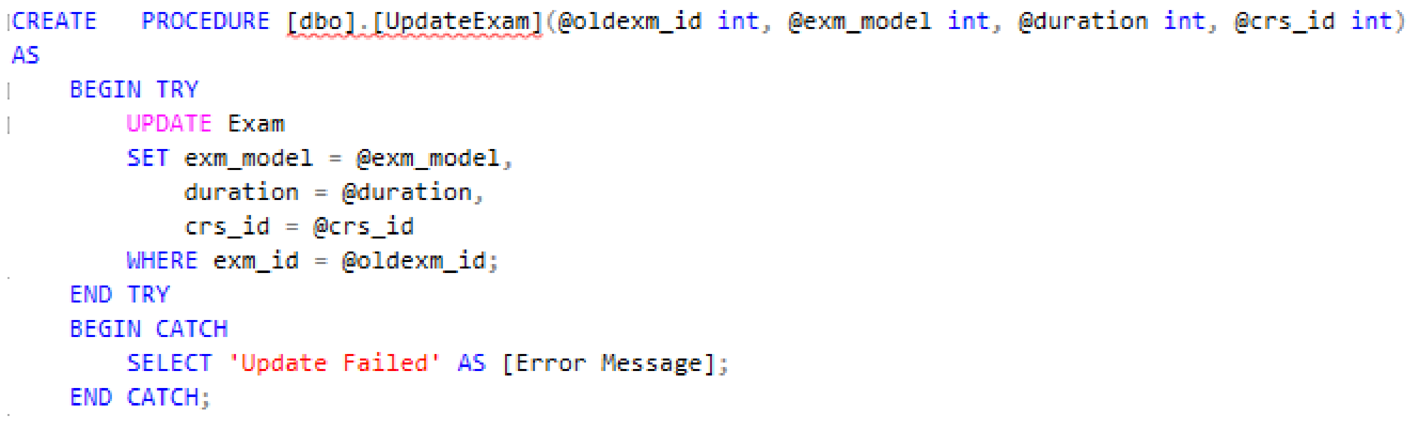
Procedure

dbo.UpdateExam

Parameter	Data Type	Default	Is Output
@oldexm_id	int		
@exm_model	int		
@duration	int		
@crs_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```
CREATE   PROCEDURE [dbo].[UpdateExam](@oldexm_id int, @exm_model int, @duration int, @crs_id int)
AS
    BEGIN TRY
        UPDATE Exam
        SET exm_model = @exm_model,
            duration = @duration,
            crs_id = @crs_id
        WHERE exm_id = @oldexm_id;
    END TRY
    BEGIN CATCH
        SELECT 'Update Failed' AS [Error Message];
    END CATCH;
```

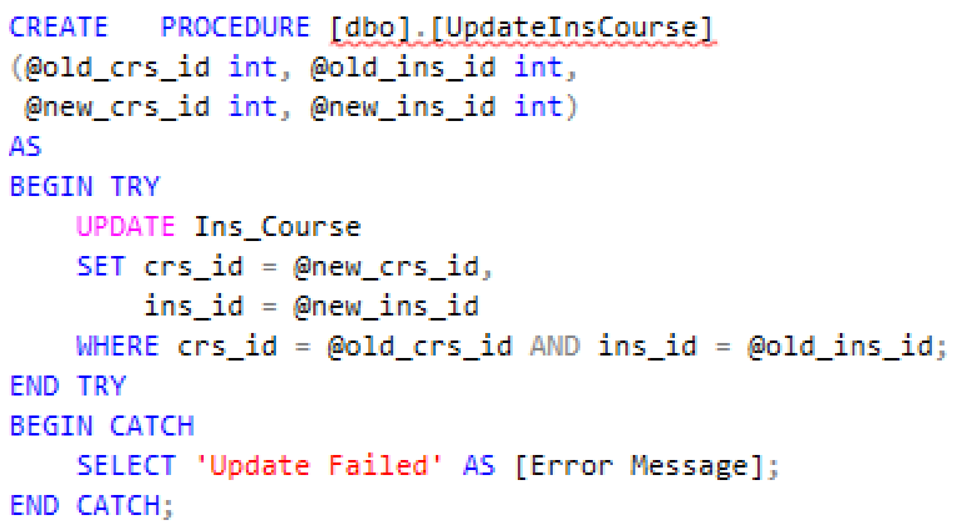
Procedure

dbo.UpdateInsCourse

Parameter	Data Type	Default	Is Output
@old_crs_id	int		
@old_ins_id	int		
@new_crs_id	int		
@new_ins_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```
CREATE   PROCEDURE [dbo].[UpdateInsCourse]
(@old_crs_id int, @old_ins_id int,
 @new_crs_id int, @new_ins_id int)
AS
BEGIN TRY
    UPDATE Ins_Course
    SET crs_id = @new_crs_id,
        ins_id = @new_ins_id
    WHERE crs_id = @old_crs_id AND ins_id = @old_ins_id;
END TRY
BEGIN CATCH
    SELECT 'Update Failed' AS [Error Message];
END CATCH;
```

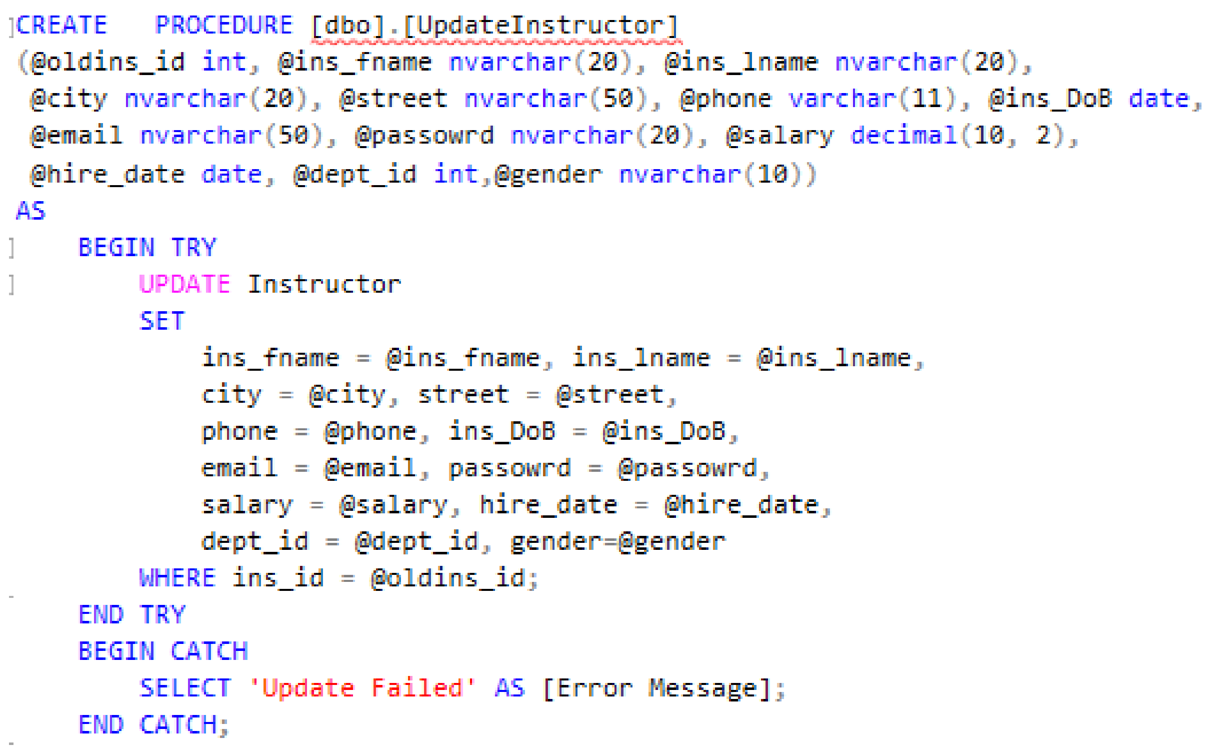
Procedure

dbo.UpdateInstructor

Parameter	Data Type	Default	Is Output
@oldins_id	int		
@ins_fname	nvarchar(20)		
@ins_lname	nvarchar(20)		
@city	nvarchar(20)		
@street	nvarchar(50)		
@phone	varchar(11)		
@ins_DoB	date		
@email	nvarchar(50)		
@passowrd	nvarchar(20)		
@salary	decimal(10,2)		
@hire_date	date		
@dept_id	int		
@gender	nvarchar(10)		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```sql
CREATE   PROCEDURE [dbo].[UpdateInstructor]
(@oldins_id int, @ins_fname nvarchar(20), @ins_lname nvarchar(20),
 @city nvarchar(20), @street nvarchar(50), @phone varchar(11), @ins_DoB date,
 @email nvarchar(50), @passowrd nvarchar(20), @salary decimal(10, 2),
 @hire_date date, @dept_id int,@gender nvarchar(10))
AS
    BEGIN TRY
        UPDATE Instructor
        SET
            ins_fname = @ins_fname, ins_lname = @ins_lname,
            city = @city, street = @street,
            phone = @phone, ins_DoB = @ins_DoB,
            email = @email, passowrd = @passowrd,
            salary = @salary, hire_date = @hire_date,
            dept_id = @dept_id, gender=@gender
        WHERE ins_id = @oldins_id;
    END TRY
    BEGIN CATCH
        SELECT 'Update Failed' AS [Error Message];
    END CATCH;
```

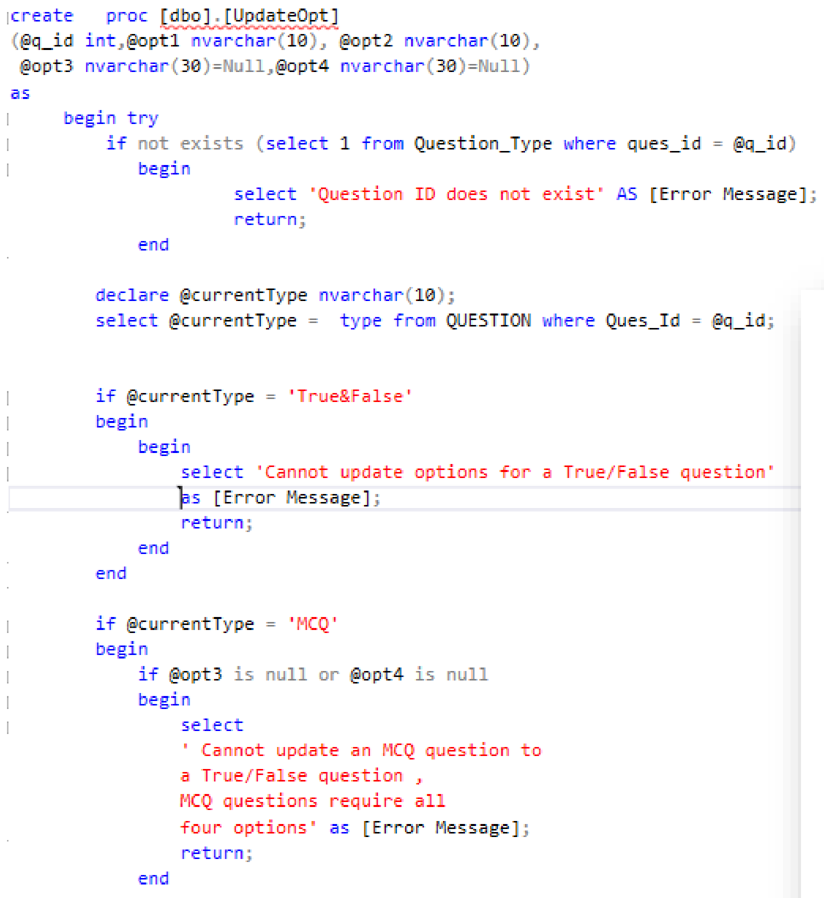
Procedure

dbo.UpdateOpt

Parameter	Data Type	Default	Is Output
@q_id	int		
@opt1	nvarchar(10)		
@opt2	nvarchar(10)		
@opt3	nvarchar(30)	Null	
@opt4	nvarchar(30)	Null	

Result:

Column	Data Type	Nullable
Error Message	varchar(26)	

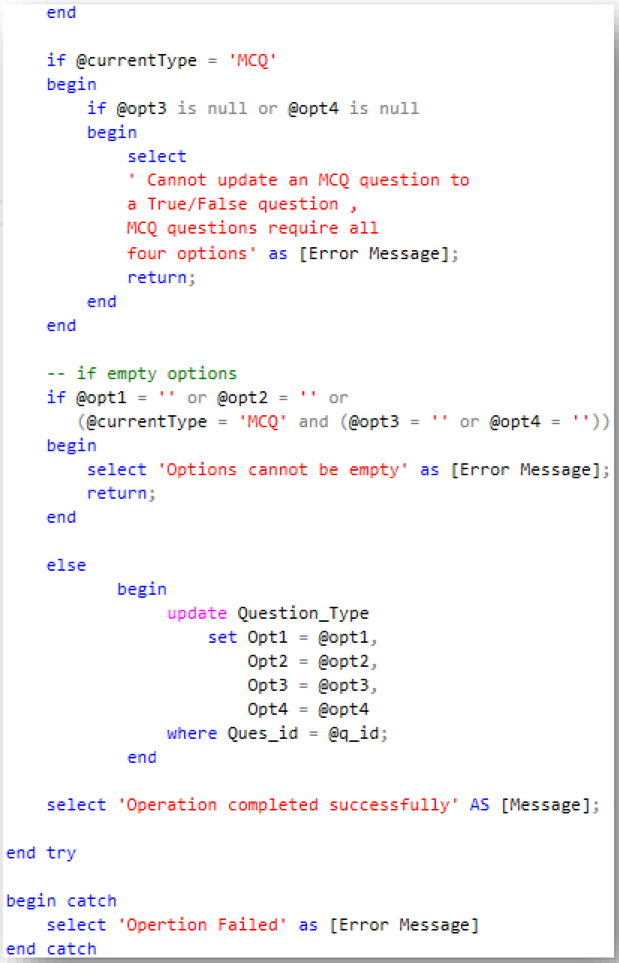


```
create   proc [dbo].[UpdateOpt]
(@q_id int,@opt1 nvarchar(10), @opt2 nvarchar(10),
 @opt3 nvarchar(30)=Null,@opt4 nvarchar(30)=Null)
as
     begin try
        if not exists (select 1 from Question_Type where ques_id = @q_id)
           begin
                    select 'Question ID does not exist' AS [Error Message];
                    return;
           end

       declare @currentType nvarchar(10);
       select @currentType =  type from QUESTION where Ques_Id = @q_id;


       if @currentType = 'True&False'
       begin
           begin
               select 'Cannot update options for a True/False question'
               as [Error Message];
               return;
           end
       end

       if @currentType = 'MCQ'
       begin
           if @opt3 is null or @opt4 is null
           begin
               select
               ' Cannot update an MCQ question to
               a True/False question ,
               MCQ questions require all
               four options' as [Error Message];
               return;
           end
```



```
    end

    if @currentType = 'MCQ'
    begin
        if @opt3 is null or @opt4 is null
        begin
            select
            ' Cannot update an MCQ question to
            a True/False question ,
            MCQ questions require all
            four options' as [Error Message];
            return;
        end
    end

    -- if empty options
    if @opt1 = '' or @opt2 = '' or
       (@currentType = 'MCQ' and (@opt3 = '' or @opt4 = ''))
    begin
        select 'Options cannot be empty' as [Error Message];
        return;
    end

    else
           begin
                update Question_Type
                    set Opt1 = @opt1,
                        Opt2 = @opt2,
                        Opt3 = @opt3,
                        Opt4 = @opt4
                where Ques_id = @q_id;
            end

    select 'Operation completed successfully' AS [Message];

end try

begin catch
    select 'Opertion Failed' as [Error Message]
end catch
```

Procedure

dbo.UpdateQuesAndOpt

Parameter	Data Type	Default	Is Output
@oldq_id	int		
@newq_id	int		
@opt1	nvarchar(10)		
@opt2	nvarchar(10)		
@opt3	nvarchar(30)	Null	
@opt4	nvarchar(30)	Null	

Result:

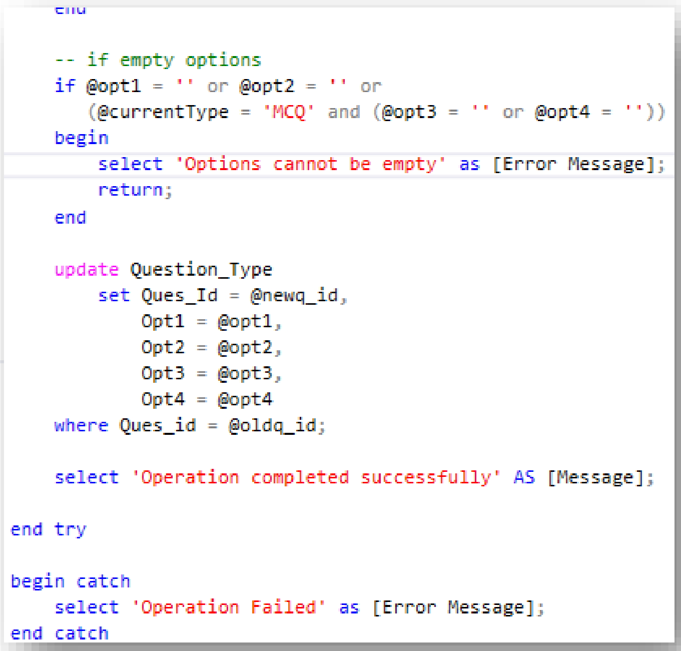
Column	Data Type	Nullable
Error Message	varchar(26)	

```sql
create   proc [dbo].[UpdateQuesAndOpt]
(@oldq_id int, @newq_id int ,@opt1 nvarchar(10),
 @opt2 nvarchar(10),@opt3 nvarchar(30)=Null,@opt4 nvarchar(30)=Null)
as
    begin try
        if not exists (select 1 from Question_Type where ques_id = @oldq_id)
        begin
            select 'Question ID does not exist' AS [Error Message];
            return;
        end
        if exists ( select 1 from Question_Type where Ques_Id=@newq_id)
        begin
            select 'The Question with new ID that you want to update already exists'
            AS [Error Message];
            return;
        end

        declare @currentType nvarchar(10);
        select @currentType =  type from QUESTION where Ques_Id = @oldq_id;


        if @currentType = 'True&False'
        begin
            if @opt3 is not null or @opt4 is not null
            begin
                select 'Cannot update options for a True/False question'
                as [Error Message];
                return;
            end
        end

        if @currentType = 'MCQ'
        begin
            if @opt3 is null or @opt4 is null
            begin
                select ' Cannot update an MCQ question to a True/False question
                ,MCQ questions require all four options' as [Error Message];
                return;
            end
```



```sql

    -- if empty options
    if @opt1 = '' or @opt2 = '' or
       (@currentType = 'MCQ' and (@opt3 = '' or @opt4 = ''))
    begin
        select 'Options cannot be empty' as [Error Message];
        return;
    end

    update Question_Type
        set Ques_Id = @newq_id,
            Opt1 = @opt1,
            Opt2 = @opt2,
            Opt3 = @opt3,
            Opt4 = @opt4
    where Ques_id = @oldq_id;

    select 'Operation completed successfully' AS [Message];

end try

begin catch
    select 'Operation Failed' as [Error Message];
end catch
```

Procedure

dbo.UpdateQuesInEx

Parameter	Data Type	Default	Is Output
@exm_id	int		
@oldq_id	int		
@newq_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(30)	

```
create   proc [dbo].[UpdateQuesInEx] (@exm_id int , @oldq_id int , @newq_id int)
as
    begin try
        if not exists (select 1 from Ques_exam where exam_id=@exm_id)
            begin
                select 'There is  no Exam With this ID' as [Error Message]
                return;
            end
        if not exists (select 1 from Ques_exam where ques_id=@oldq_id and exam_id=@exm_id)
            begin
                select 'there is no question with this ID in this Exam to Update' as [Message]
                return;
            end

        update Ques_exam
        set ques_id =@newq_id
        where exam_id =@exm_id and ques_id=@oldq_id

        select 'Update Operation completed successfully' AS [Message];
    end try
    begin catch
        select 'Update Failed' as [Error Message]
    end catch
```

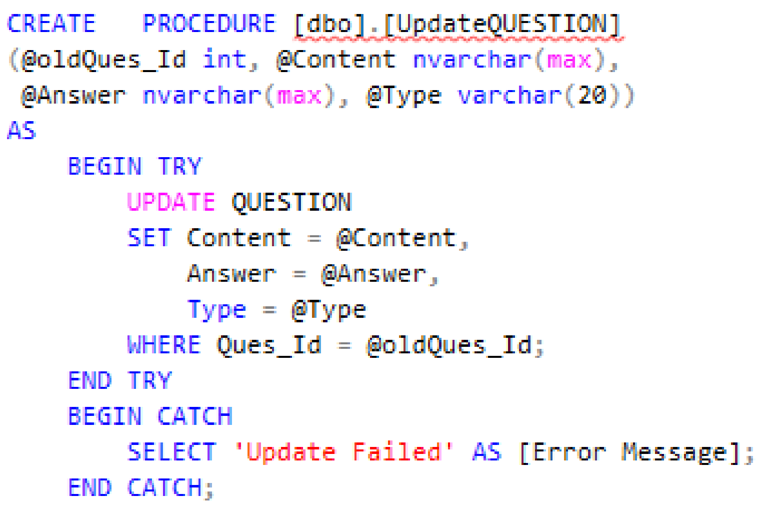
Procedure

dbo.UpdateQUESTION

Parameter	Data Type	Default	Is Output
@oldQues_Id	int		
@Content	nvarchar(max)		
@Answer	nvarchar(max)		
@Type	varchar(20)		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```
CREATE   PROCEDURE [dbo].[UpdateQUESTION]
(@oldQues_Id int, @Content nvarchar(max),
 @Answer nvarchar(max), @Type varchar(20))
AS
    BEGIN TRY
        UPDATE QUESTION
        SET Content = @Content,
            Answer = @Answer,
            Type = @Type
        WHERE Ques_Id = @oldQues_Id;
    END TRY
    BEGIN CATCH
        SELECT 'Update Failed' AS [Error Message];
    END CATCH;
```

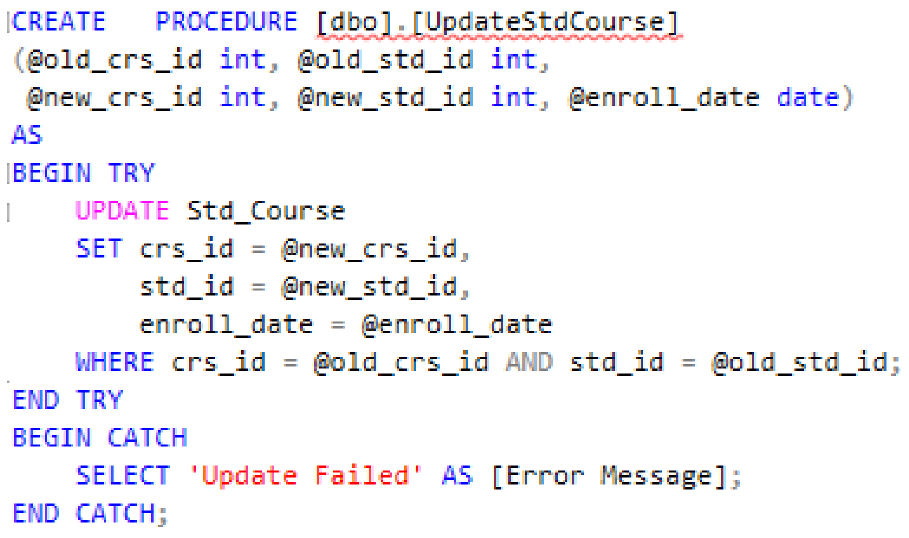
Procedure

dbo.UpdateStdCourse

Parameter	Data Type	Default	Is Output
@old_crs_id	int		
@old_std_id	int		
@new_crs_id	int		
@new_std_id	int		
@enroll_date	date		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```
CREATE   PROCEDURE [dbo].[UpdateStdCourse]
(@old_crs_id int, @old_std_id int,
 @new_crs_id int, @new_std_id int, @enroll_date date)
AS
BEGIN TRY
    UPDATE Std_Course
    SET crs_id = @new_crs_id,
        std_id = @new_std_id,
        enroll_date = @enroll_date
    WHERE crs_id = @old_crs_id AND std_id = @old_std_id;
END TRY
BEGIN CATCH
    SELECT 'Update Failed' AS [Error Message];
END CATCH;
```

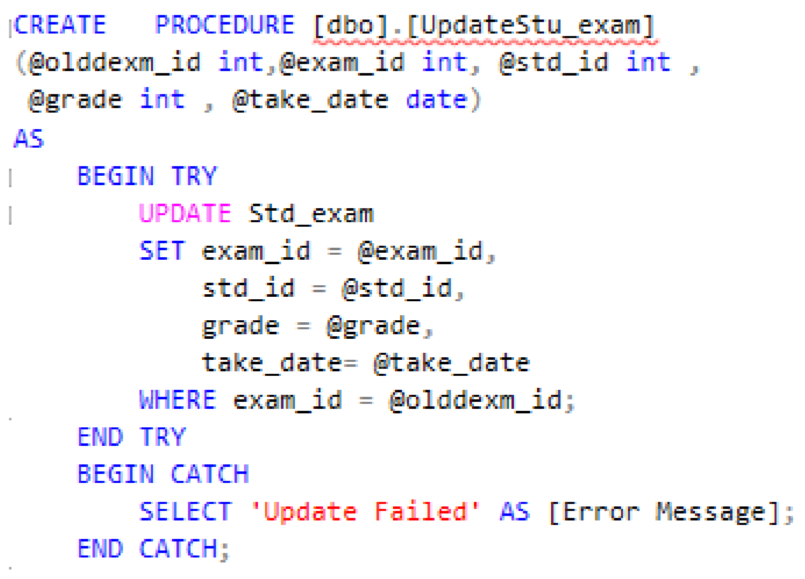
Procedure

dbo.UpdateStu_exam

Parameter	Data Type	Default	Is Output
@olddexm_id	int		
@exam_id	int		
@std_id	int		
@grade	int		
@take_date	date		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	



```
CREATE   PROCEDURE [dbo].[UpdateStu_exam]
(@olddexm_id int,@exam_id int, @std_id int ,
 @grade int , @take_date date)
AS
    BEGIN TRY
        UPDATE Std_exam
        SET exam_id = @exam_id,
            std_id = @std_id,
            grade = @grade,
            take_date= @take_date
        WHERE exam_id = @olddexm_id;
    END TRY
    BEGIN CATCH
        SELECT 'Update Failed' AS [Error Message];
    END CATCH;
```

Procedure

dbo.UpdateStudent

Parameter	Data Type	Default	Is Output
@oldstd_id	int		
@std_fname	nvarchar(20)		
@std_lname	nvarchar(20)		
@city	nvarchar(20)		
@street	nvarchar(50)		
@phone	varchar(11)		
@std_DoB	date		
@email	nvarchar(50)		
@passowrd	nvarchar(20)		
@dept_id	int		
@gneder	nvarchar(10)		

Result:

Column	Data Type	Nullable
Error Message	nvarchar(4000)	X

```
CREATE   PROCEDURE [dbo].[UpdateStudent]
(@oldstd_id int,  @std_fname nvarchar(20), @std_lname nvarchar(20),
 @city nvarchar(20), @street nvarchar(50), @phone varchar(11),
 @std_DoB date, @email nvarchar(50), @passowrd nvarchar(20),
 @dept_id int, @gneder nvarchar(10))
AS
    BEGIN TRY
        UPDATE Student
        SET
            std_fname = @std_fname, std_lname = @std_lname,
            city = @city, street = @street,
            phone = @phone, std_DoB = @std_DoB,
            email = @email, passowrd = @passowrd,
            dept_id = @dept_id,
            gender=@gneder
        WHERE std_id = @oldstd_id;
    END TRY
    BEGIN CATCH
        SELECT ERROR_MESSAGE() AS [Error Message];
    END CATCH;
```

Procedure

dbo.UpdateTopic

Parameter	Data Type	Default	Is Output
@oldtop_id	int		
@top_name	nvarchar(50)		
@crs_id	int		

Result:

Column	Data Type	Nullable
Error Message	varchar(13)	

```
CREATE   PROCEDURE [dbo].[UpdateTopic](@oldtop_id int, @top_name nvarchar(50), @crs_id int)
AS
    BEGIN TRY
        UPDATE Topic
        SET top_name = @top_name,
            crs_id = @crs_id
        WHERE top_id = @oldtop_id;
    END TRY
    BEGIN CATCH
        SELECT 'Update Failed' AS [Error Message];
    END CATCH;
```

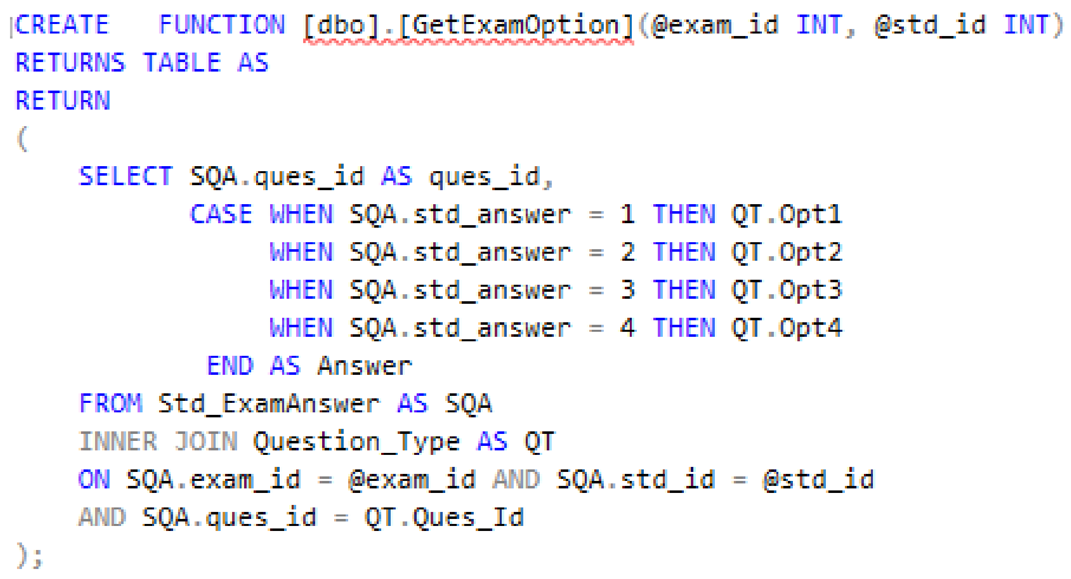
Functions:

Inline Table Valued Function dbo.GetExamOption

Parameter	Data Type	Default
@exam_id	int	
@std_id	int	

Result:

Column	Data Type	Nullable
ques_id	int	
Answer	nvarchar(100)	X



```
CREATE   FUNCTION [dbo].[GetExamOption](@exam_id INT, @std_id INT)
RETURNS TABLE AS
RETURN
(
    SELECT SQA.ques_id AS ques_id,
           CASE WHEN SQA.std_answer = 1 THEN QT.Opt1
                WHEN SQA.std_answer = 2 THEN QT.Opt2
                WHEN SQA.std_answer = 3 THEN QT.Opt3
                WHEN SQA.std_answer = 4 THEN QT.Opt4
            END AS Answer
    FROM Std_ExamAnswer AS SQA
    INNER JOIN Question_Type AS QT
    ON SQA.exam_id = @exam_id AND SQA.std_id = @std_id
    AND SQA.ques_id = QT.Ques_Id
);
```

Reports

Course Name: C#   Exam Model: 3

1- Is CSS used to style webpages?

- ◯ True
- ◯ False

2- Is 15 > 20?

- ◯ True
- ◯ False

3- What does CSS stand for?

◯ Cascading Style Sheets	◯ Cascading Simple Sheets
◯ Cascading Scripting Style	◯ Cascading Syntax Styles

4- Which data structure is used for implement...

- ◯ Stack
- ◯ Queue

5- Is JavaScript the same as Java?

- ◯ True
- ◯ False

6- Which of these is an example of an interpreted langua...

◯ Python ◯ Java

◯ C++ ◯ Ruby

7- Which of the following is the best sorting algorithm for large datasets?

◯ Merge Sort ◯ Bubble Sort

◯ Quick Sort ◯ Insertion Sort

8- Is Swift a modern programming language?

◯ True

◯ False

9- Is the Python "list" mutable?

◯ True

◯ False

10- Is the `print()` function in Python used to output data to t...

◯ True

◯ False

Course Name: RWD **Student Name:** Ahmed Ali

Question Content	Student Answer
Is 5 > 3?	True
Is C# primarily used for web development?	False
Is C++ a procedural language?	True
Is CSS used to style webpages?	True
Is HTML used for creating webpages?	False
Is JavaScript the same as Java?	False
Is Python a programming language?	True
Is Python interpreted or compiled?	Compiled
Is Swift a modern programming language?	True
Is the `print()` function in Python used to output data to the console?	True
What is a static variable in C programming?	A variable that is only local to the function
What is the full form of IDE?	Intelligent Development Engine
Which is the main advantage of using recursion in programming?	Requires more memory
Which language is used for web development?	HTML
Which of the following is a key characteristic of functional programming?	Recursion
Which of the following is a programming language?	Java
Which of the following is an example of a functional programming language?	Haskell

Courses and Enrolled Students Number to Instructor: Tamer Salem

Course Name	Number Enrolled Students
C#	4
EF	3

Grades in all exams courses for the student: Ahmed Ali

Course Name	Course ID ▲	Student Grade
HTML	1	40
RWD	4	52

Data of Students are Enrolled in Department: UX

Sudent ID ▲	Fname	Lname	City	Street	Year	Month	Day	Phone	Email	Passowrd	gender
6	Rana	Ibrahim	Minya	Street 6	2003	April	19	1269810235	rana.ibrahim223@example.com	password6	F
16	Salma	Ibrahim	Minya	Street 16	1999	August	24	1130011006	salma.ibrahim116@example.com	password16	F
26	Samah	Ibrahim	Minya	Street 26	2000	April	14	153011016	samah.ibrahim126@example.com	password26	F
36	Mahmoud	Khaled	Minya	Street 36	2003	March	11	1131111026	mahmoud.khaled136@example.com	password36	M
42	Dina	Samy	Giza	Street 43	1999	September	21	1212345678	dina.samy43@example.com	password43	F
48	Fady	Gad	Mansoura	Street 49	2003	April	10	1156789012	fady.gad49@example.com	password49	M
58	Yara	Sami	Cairo	Street 59	2003	March	1	1298765432	yara.sami59@example.com	password59	F

Topics in course: HTML

Topic ID ▲	Topic Name
1	tags
2	links
3	imgs
4	table
5	forms
6	lists
7	colors
8	styles