

3/6/2025

**Supervisor: Eng. Marwa**

Examination System



**Team Members:**

1. **Santy Osama Mina**
2. **Yousry Essam Ayoub**



***ITI Intake 45, Assiut Branch***

Santy Osama& Yousry Essam

**Table Of Contents**

Contents

[**Abstract** 2](#_Toc192277845)

[**Database Design** 3](#_Toc192277846)

[1.1 Entity-Relationship Diagram (ERD): 3](#_Toc192277847)

[1.1.1Overview: 3](#_Toc192277848)

[1.1.2 ERD Diagram Representation: 5](#_Toc192277849)

[1.2 Database Schema Design 6](#_Toc192277850)

[1.3 Database Dictionary: 9](#_Toc192277851)

[**Stored Procedure:** 13](#_Toc192277852)

[**1.Insertion:** 13](#_Toc192277853)

[**2.Deletion:** 19](#_Toc192277854)

[**3.Exam Generation:** 23](#_Toc192277855)

[**4.Exam Correction:** 24](#_Toc192277856)

[**Views:** 26](#_Toc192277857)

[**Reports:** 27](#_Toc192277858)

[**Data Base Backup:** 33](#_Toc192277859)

# **Abstract**

In the modern educational landscape, online examination systems have become essential for ensuring seamless and efficient assessment processes. This project aims to develop an **Automated Online Examination System** that facilitates the creation, execution, and evaluation of online exams. The system is supported by a **structured SQL database**, ensuring data integrity and efficient query performance.

To achieve this, an **Entity-Relationship Diagram (ERD)** was designed to define the relationships between various entities such as students, courses, exams, and questions. Additionally, a **Database Dictionary** was created to document table structures, attributes, and constraints.

The system also includes **stored procedures** to handle core functionalities:

* **Basic CRUD operations** (Select, Insert, Update, Delete) on all tables.
* **Exam Generation** to randomly create exams with different types of questions.
* **Exam Answers Management** to store students' responses.
* **Exam Correction** to automate grading based on predefined correct answers.
* This automated system enhances the efficiency of exam administration, reduces manual effort, and ensures fairness in assessment. Future enhancements may include **reporting tools (SSRS, Power BI)** and **integration with social media platforms** to further expand system capabilities.

# **Database Design**

## Entity-Relationship Diagram (ERD):

### 1.1.1Overview:

The **Entity-Relationship Diagram (ERD)** represents the structure of the **Online Examination System** by illustrating the relationships between various tables/entities. This system manages **students, instructors, courses, exams, questions, answers, branches, and tracks** while ensuring **data integrity and normalization**.

**Key Entities & Relationships**

1. **Branch**
   * A **Branch** offers multiple **Tracks**.
   * Relationship: **One-to-Many** (Branch → Track).
2. **Instructor**
   * An **Instructor** manages a **Track**.
   * Relationship: **One-to-One** (Instructor → Track).
3. **Track**
   * A **Track** contains multiple **Courses**.
   * Relationship: **One-to-Many** (Track → Course).
4. **Student**
   * A **Student** is assigned to one **Track**.
   * A **Student** enrolls in multiple **Courses**.
   * Relationship: **Many-to-One** (Student → Track), **Many-to-Many** (Student ↔ Course).
5. **Course**
   * A **Course** is associated with multiple **Exams**.
   * A **Course** contains multiple **Questions**.
   * Relationship: **One-to-Many** (Course → Exam), **One-to-Many** (Course → Question).
6. **Exam**
   * An **Exam** consists of multiple **Questions**.
   * A **Student** takes multiple **Exams**.
   * Relationship: **Many-to-Many** (Exam ↔ Question), **Many-to-Many** (Student ↔ Exam).
7. **Question**
   * A **Question** has multiple **Options**.
   * Relationship: **One-to-Many** (Question → Option).
8. **Option**
   * Each **Option** belongs to one **Question** and indicates whether it is correct.
   * Relationship: **Many-to-One** (Option → Question).
9. **Student\_Exam\_Question**
   * Stores **student answers** to exam questions.
   * Relationship: **Many-to-Many** (Student ↔ Question in an Exam).

### 1.1.2 ERD Diagram Representation:

A diagram of a company

AI-generated content may be incorrect.

## Database Schema Design

The **Database Schema Design** outlines the structure of the **Online Examination System** database. It includes tables, their attributes, relationships, and constraints to ensure data integrity and efficiency.

**Schema Overview**

The database consists of **13 tables**, each serving a specific purpose in managing **students, courses, exams, questions, and results**. Below is an overview of the schema design:

1. **Branch** – Stores branch details.
2. **Instructor** – Stores instructors' details.
3. **Track** – Defines different study tracks.
4. **Course** – Contains course information.
5. **Student** – Manages student records.
6. **Exam** – Stores exam details.
7. **Question** – Contains exam questions.
8. **Option\_Table** – Holds multiple-choice question options.
9. **Branch\_Track** – Manages branch and track relationships.
10. **Track\_Course** – Defines courses assigned to tracks.
11. **Student\_Course** – Tracks students' enrollment and scores in courses.
12. **Student\_Exam\_Question** – Stores students' answers to exam questions.
13. **Exam\_Question** – Links exams to their questions.

**Schema Constraints**

* **Primary Keys (PK):** Ensure unique identification of each record.
* **Foreign Keys (FK):** Establish relationships between tables.
* **Unique Constraints:** Prevent duplicate values where necessary.
* **Check Constraints:** Maintain data validity (e.g., non-negative salaries, valid scores).

**Relationships Between Tables**

* **Branch ↔ Track**: A branch can have multiple tracks (Branch\_Track table).
* **Track ↔ Course**: A track can offer multiple courses (Track\_Course table).
* **Course ↔ Exam**: Each course can have multiple exams.
* **Exam ↔ Question**: Each exam consists of multiple questions.
* **Question ↔ Option\_Table**: Each question has multiple options.
* **Student ↔ Track**: A student belongs to a specific track.
* **Student ↔ Course**: A student enrolls in multiple courses (Student\_Course table).
* **Student ↔ Exam**: Students take exams and answer questions (Student\_Exam\_Question table).

This schema ensures a **normalized database** with **efficient data retrieval, consistency, and integrity** for the **Online Examination System**.

**Database Mapping**

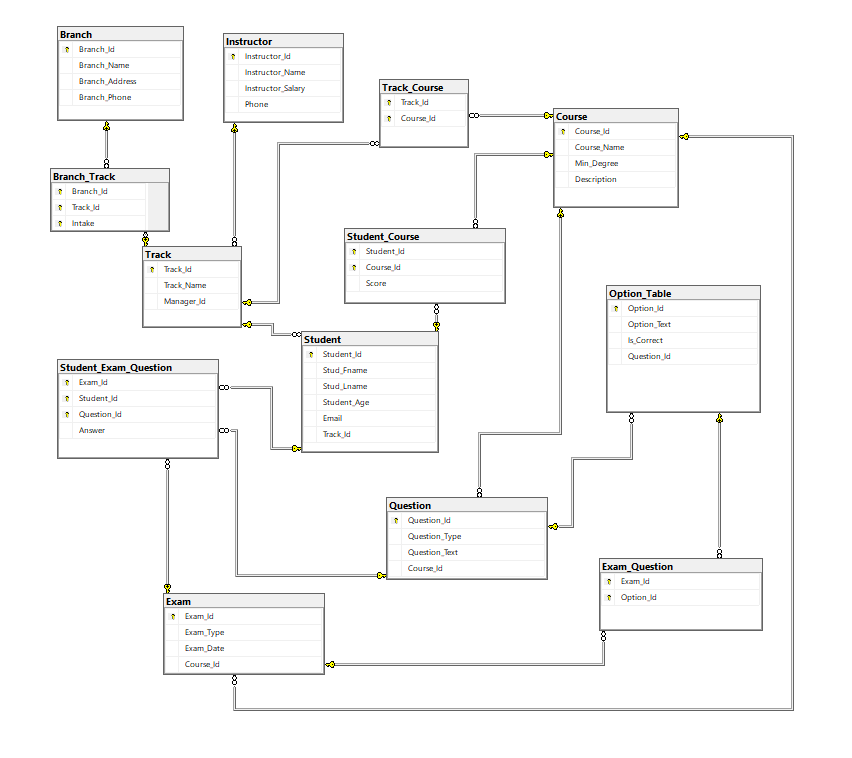
A screenshot of a computer

AI-generated content may be incorrect.

Link Of ERD and Mapping:

[Flowchart - Miro](https://miro.com/app/board/uXjVIX5VOOM=/)

**Database Diagram:**



## 1.3 Database Dictionary:

#### **Overview:**

The **Database Dictionary** defines the structure of the database by listing all tables along with their attributes, data types, constraints, and a brief description of each column. This ensures clarity in understanding the **schema design** and **data integrity constraints**.

**1. Branch Table:**

Stores information about different training branches.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| branch\_id | INT | PRIMARY KEY | Unique identifier for each branch |
| branch\_name | NVARCHAR(50) | NOT NULL | Name of the branch |
| branch\_address | NVARCHAR(100) | NOT NULL | Branch location |
| branch\_phone | NVARCHAR(15) | NULLABLE | Contact number of the branch |

**2. Instructor Table:**

Stores details of instructors.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| instructor\_id | INT | PRIMARY KEY | Unique identifier for instructors |
| instructor\_name | VARCHAR(50) | NOT NULL | Instructor's full name |
| instructor\_salary | DECIMAL(10,2) | CHECK (instructor\_salary >= 0) | Salary of the instructor |
| phone | VARCHAR(15) | UNIQUE | Instructor’s contact number |

**3. Track Table:**

Defines different study tracks in the system.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| track\_id | INT | PRIMARY KEY | Unique identifier for each track |
| track\_name | VARCHAR(50) | NOT NULL | Name of the track |
| manager\_id | INT | FOREIGN KEY | References Instructor(instructor\_id) |

**4. Course Table:**

Stores details of courses.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| course\_id | INT | PRIMARY KEY | Unique identifier for courses |
| course\_name | VARCHAR(50) | NOT NULL | Name of the course |
| min\_degree | INT | CHECK (min\_degree >= 0) | Minimum passing grade |
| description | TEXT | NULLABLE | Course description |

**5. Student Table:**

Stores student information.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| student\_id | INT | PRIMARY KEY | Unique identifier for students |
| stud\_fname | VARCHAR(50) | NOT NULL | First name of the student |
| stud\_lname | VARCHAR(50) | NOT NULL | Last name of the student |
| student\_age | INT | CHECK (student\_age > 0) | Age of the student |
| email | VARCHAR(100) | UNIQUE, NOT NULL | Student email for login |
| track\_id | INT | FOREIGN KEY | References Track(track\_id) |

**6. Exam Table:**

Stores details about exams.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| exam\_id | INT | PRIMARY KEY | Unique identifier for each exam |
| exam\_type | VARCHAR(50) | NOT NULL | Type of exam (e.g., Online, Written) |
| exam\_date | DATE | DEFAULT GETDATE() | Date when the exam is scheduled |
| course\_id | INT | FOREIGN KEY | References Course(course\_id) |

**7. Question Table:**

Stores questions for exams.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| question\_id | INT | PRIMARY KEY | Unique identifier for each question |
| question\_type | VARCHAR(50) | CHECK (question\_type IN ('MCQ', 'MMCQ', 'T/F')) | Type of question |
| question\_text | TEXT | NOT NULL | Content of the question |
| course\_id | INT | FOREIGN KEY | References Course(course\_id) |

**8. Option\_Table:**

Stores answer choices for questions.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| option\_id | INT | PRIMARY KEY | Unique identifier for each option |
| option\_text | TEXT | NOT NULL | Answer choice text |
| is\_correct | BIT | NOT NULL | 1 if correct, 0 if incorrect |
| question\_id | INT | FOREIGN KEY | References Question(question\_id) |

**9. Branch\_Track Table:**

Stores the relationship between branches and tracks.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| branch\_id | INT | PRIMARY KEY, FOREIGN KEY | References Branch(branch\_id) |
| track\_id | INT | PRIMARY KEY, FOREIGN KEY | References Track(track\_id) |
| intake | INT | PRIMARY KEY | The intake number for this track |

**10. Track\_Course Table**

Defines courses assigned to each track.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| track\_id | INT | PRIMARY KEY, FOREIGN KEY | References Track(track\_id) |
| course\_id | INT | PRIMARY KEY, FOREIGN KEY | References Course(course\_id) |

**11. Student\_Course Table**

Tracks student enrollment and scores in courses.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| student\_id | INT | PRIMARY KEY, FOREIGN KEY | References Student(student\_id) |
| course\_id | INT | PRIMARY KEY, FOREIGN KEY | References Course(course\_id) |
| score | DECIMAL(5,2) | CHECK (score BETWEEN 0 AND 100) | Student's score |

**12. Student\_Exam\_Question Table:**

Stores students' answers to exam questions.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| exam\_id | INT | PRIMARY KEY, FOREIGN KEY | References Exam(exam\_id) |
| student\_id | INT | PRIMARY KEY, FOREIGN KEY | References Student(student\_id) |
| question\_id | INT | PRIMARY KEY, FOREIGN KEY | References Question(question\_id) |
| answer | TEXT | NOT NULL | Student's submitted answer |

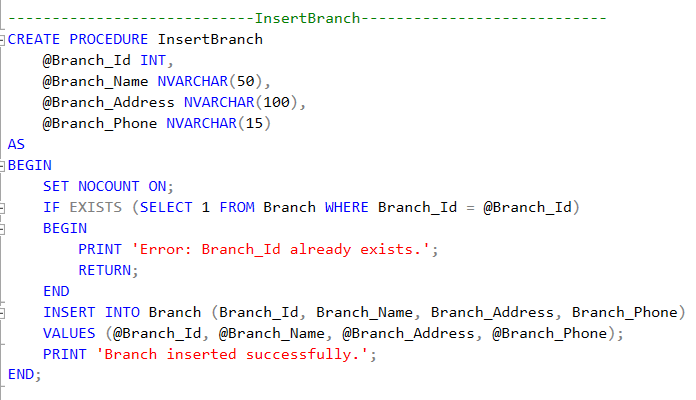
**13. Exam\_Question Table:**  
Links exams with their respective questions.

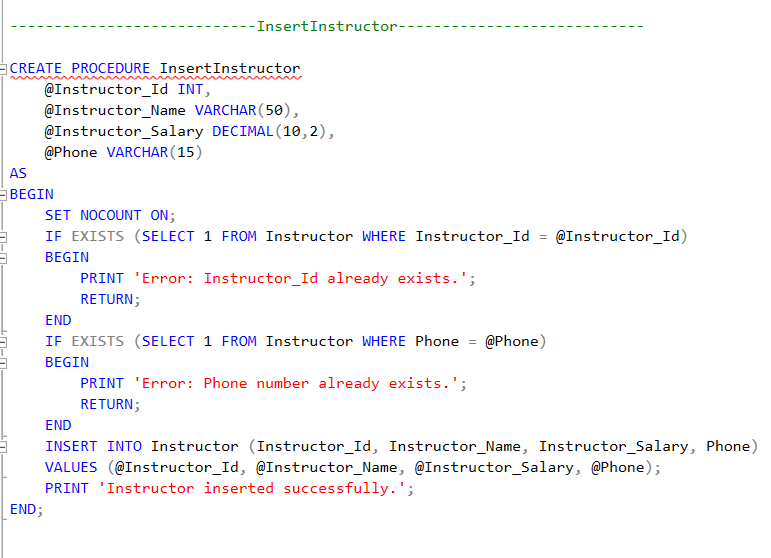
|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraints | Description |
| exam\_id | INT | PRIMARY KEY, FOREIGN KEY | References Exam(exam\_id) |
| option\_id | INT | PRIMARY KEY, FOREIGN KEY | References Option\_Table(option\_id) |

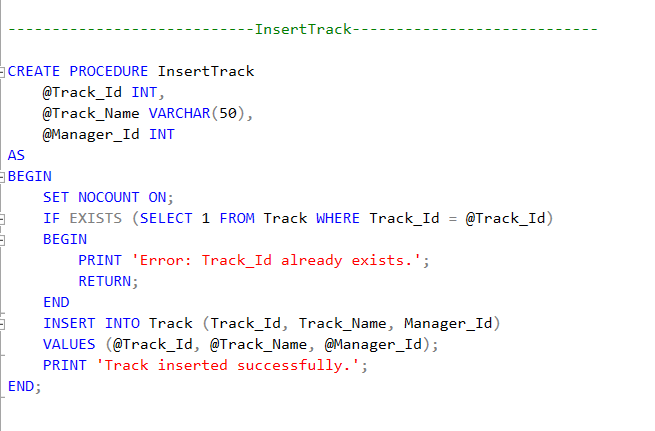
This **Database Dictionary** ensures a well-structured relational database design while maintaining **data integrity** and **efficiency**.

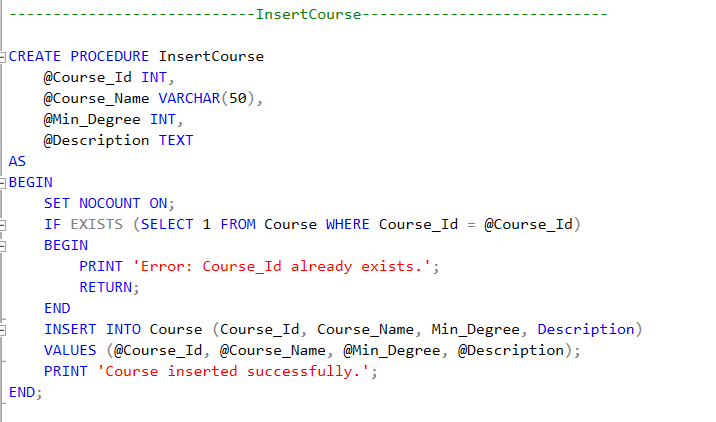
# **Stored Procedure:**

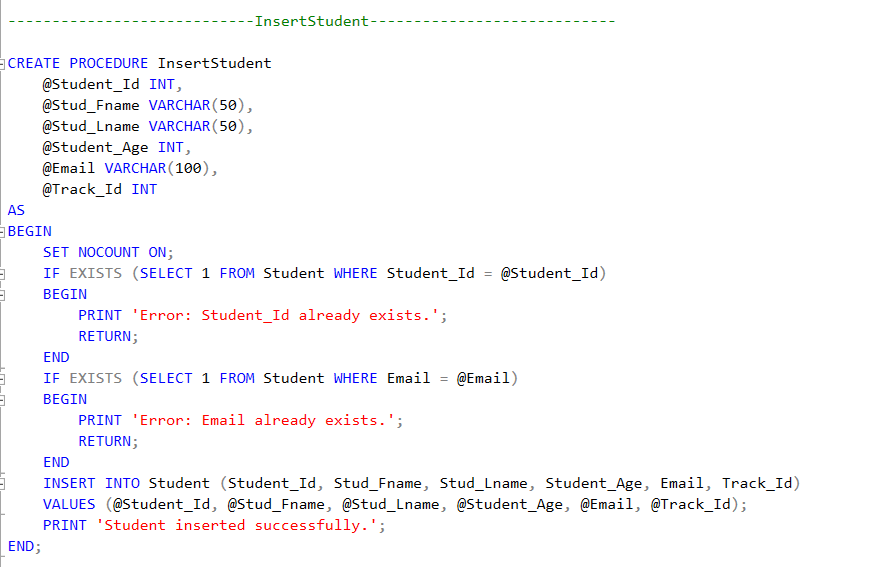
## **1.Insertion:**

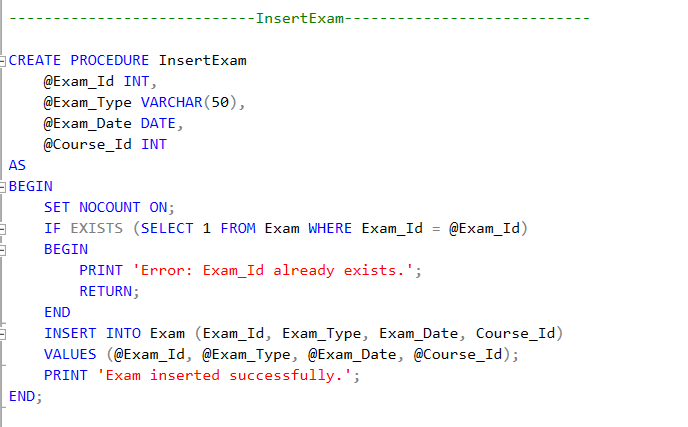


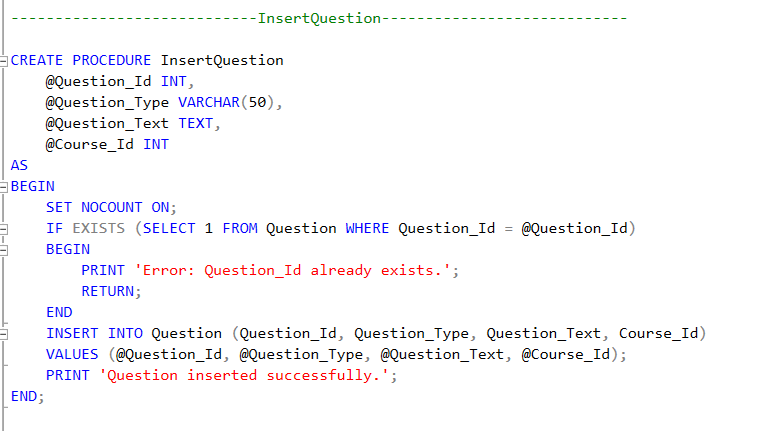


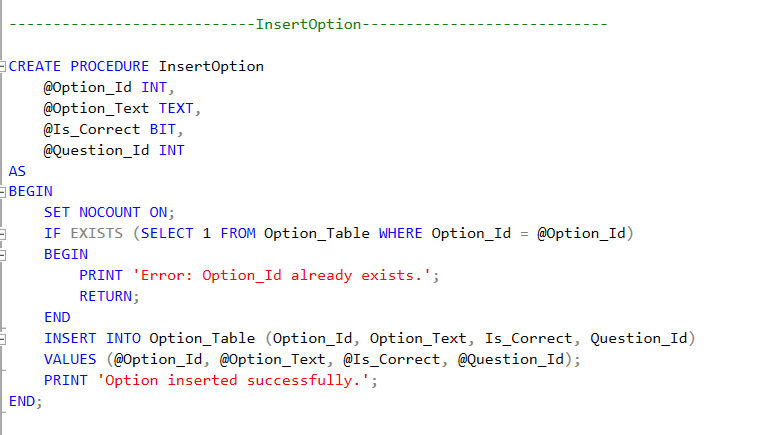


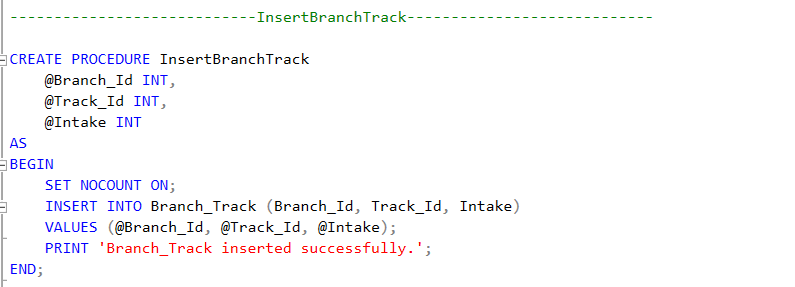


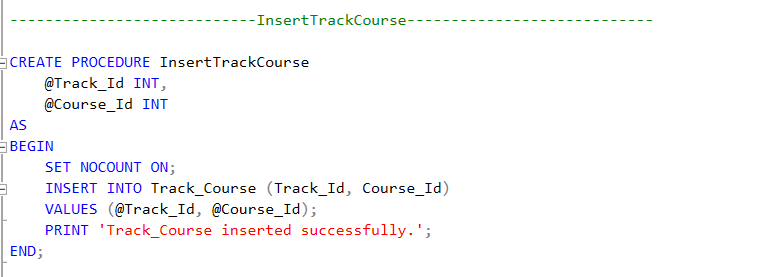


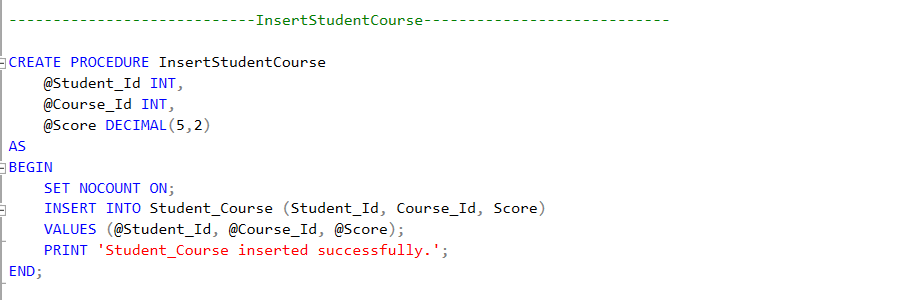


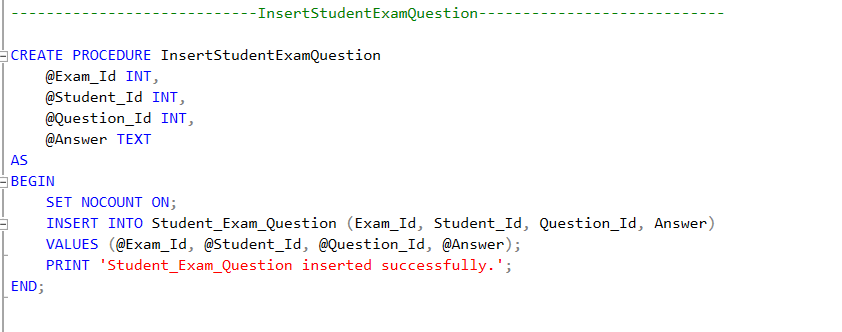


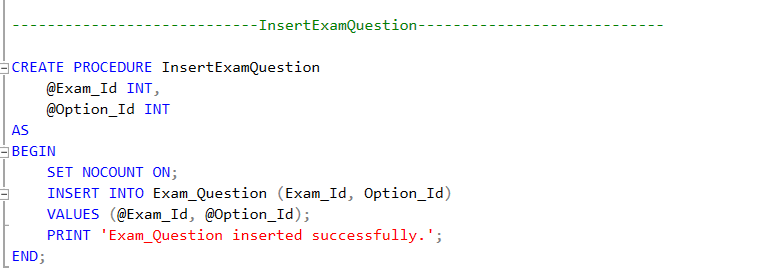




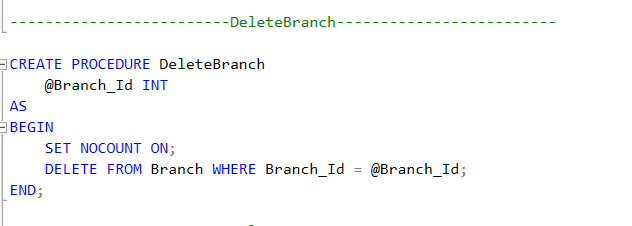


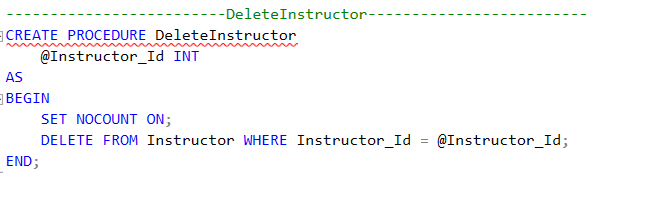


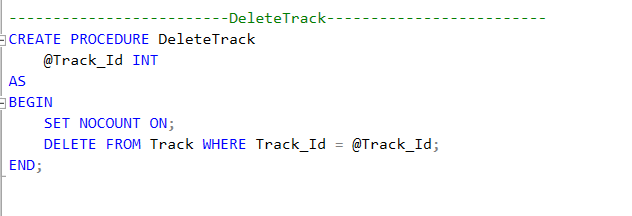


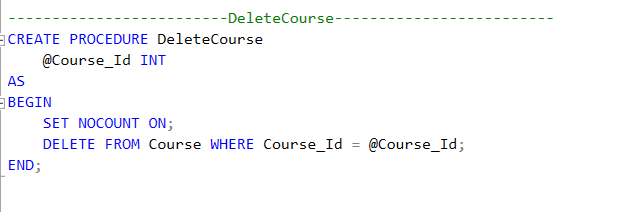


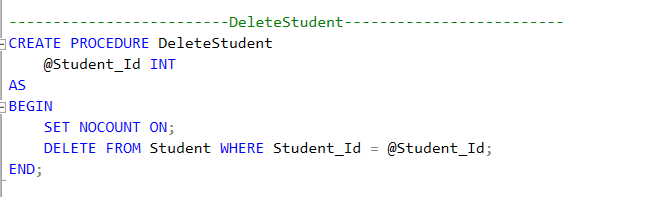
## **2.Deletion:**

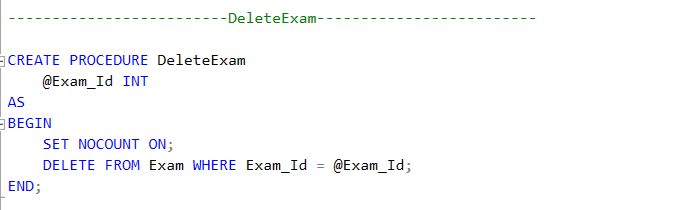


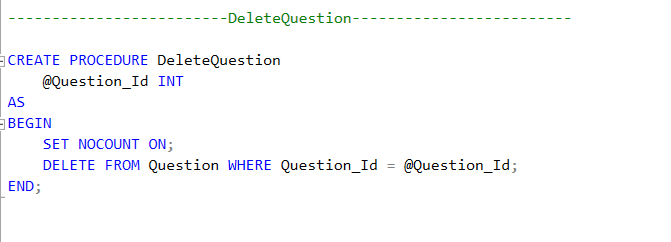


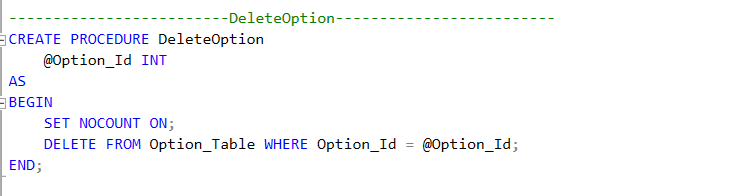


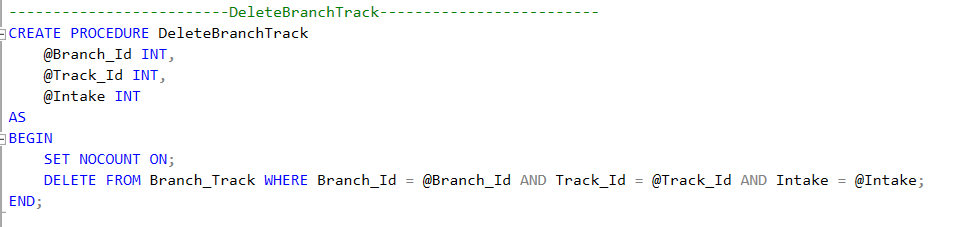


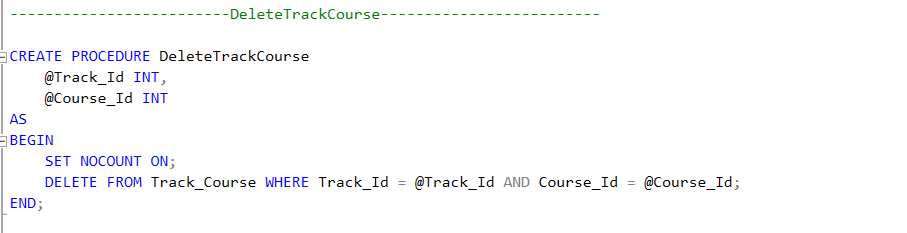


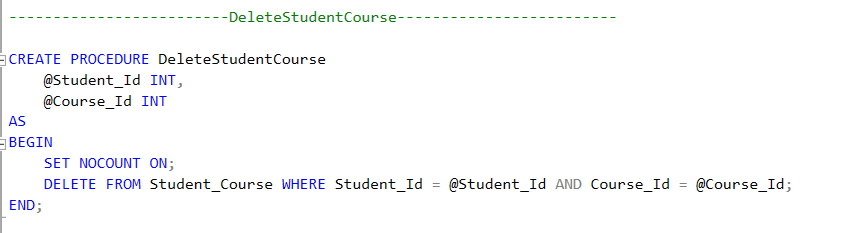


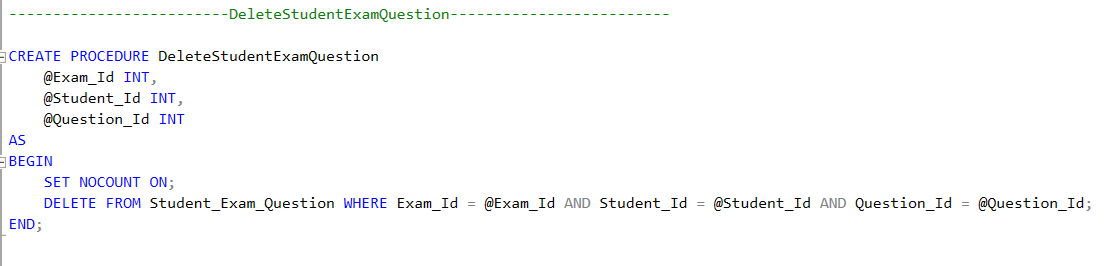


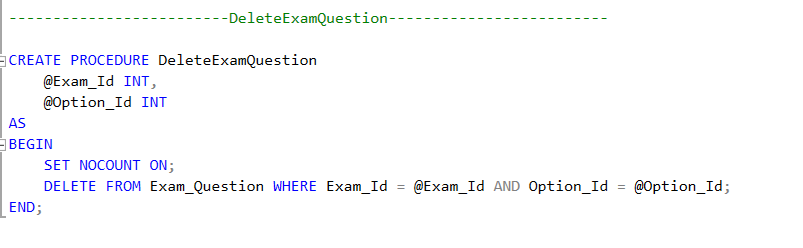




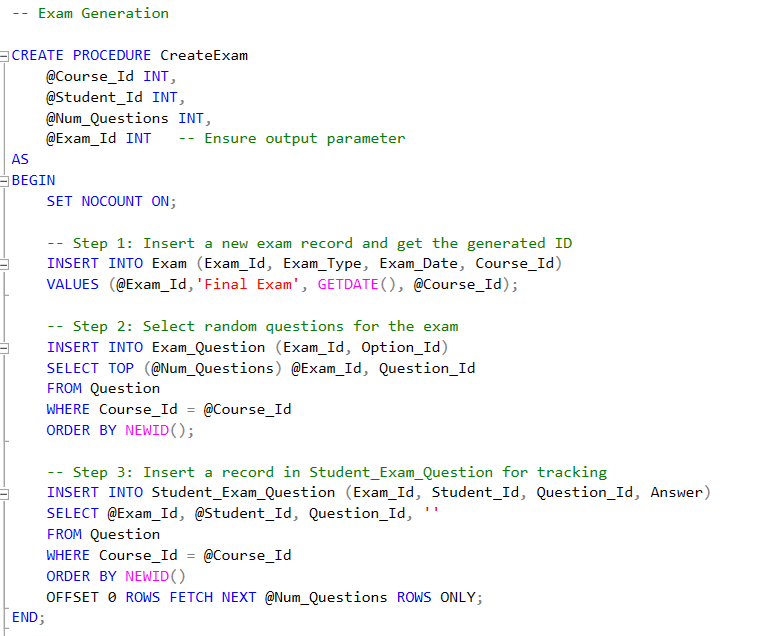




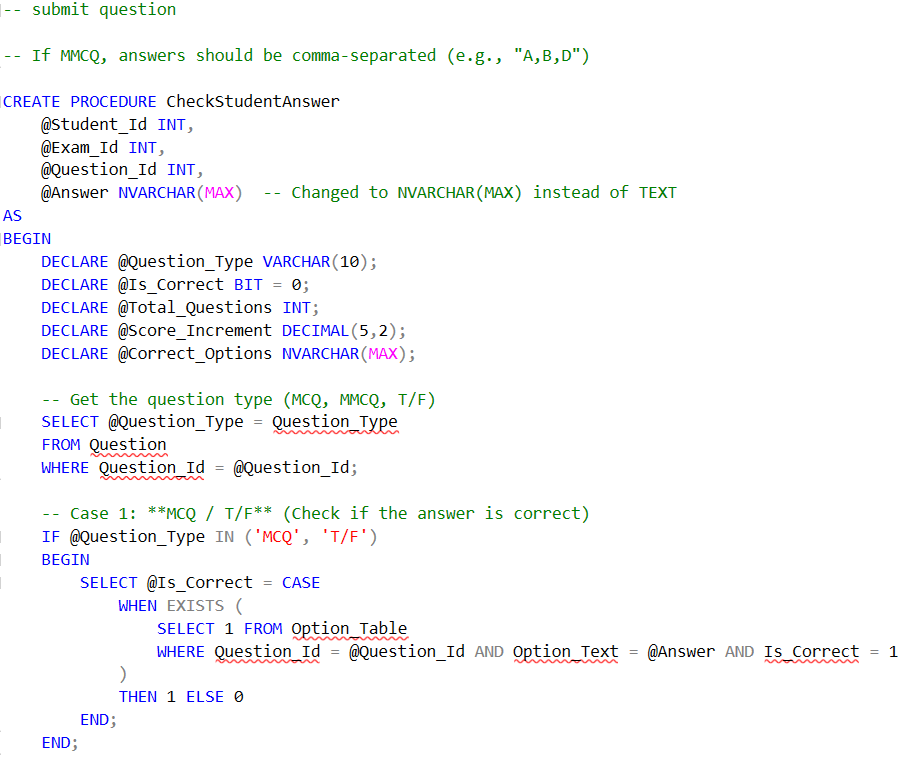


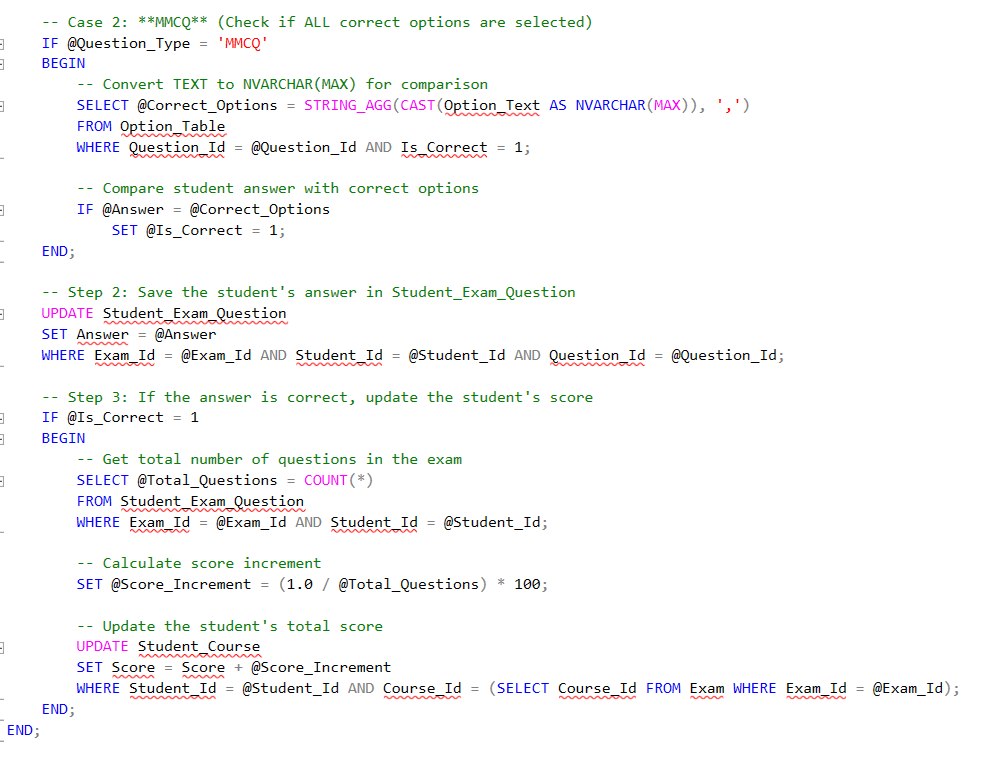


## **3.Exam Generation:**

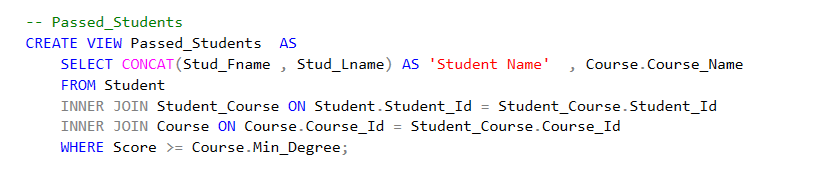


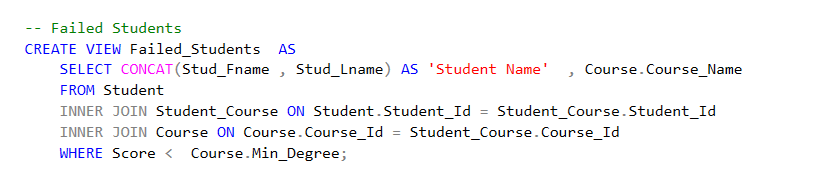
## **4.Exam Correction:**

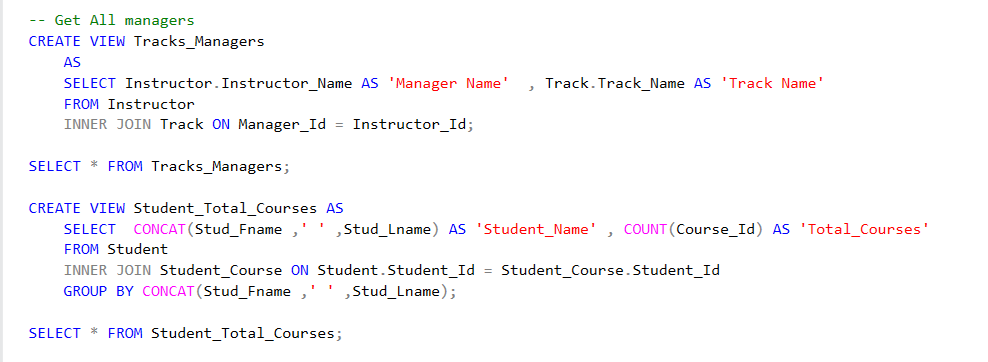




# **Views:**



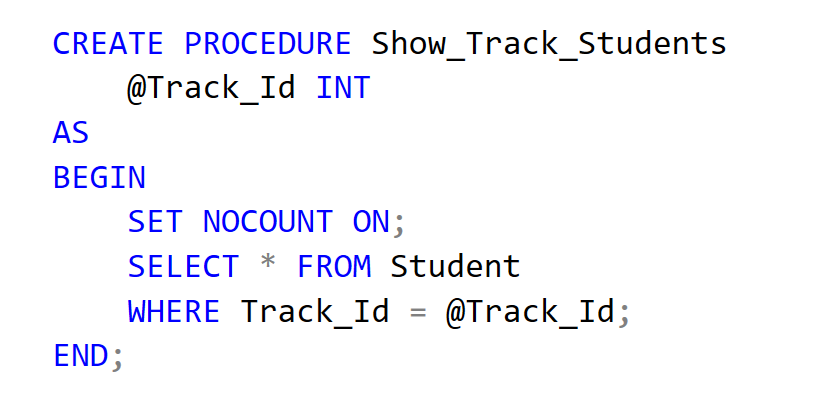




# **Reports:**

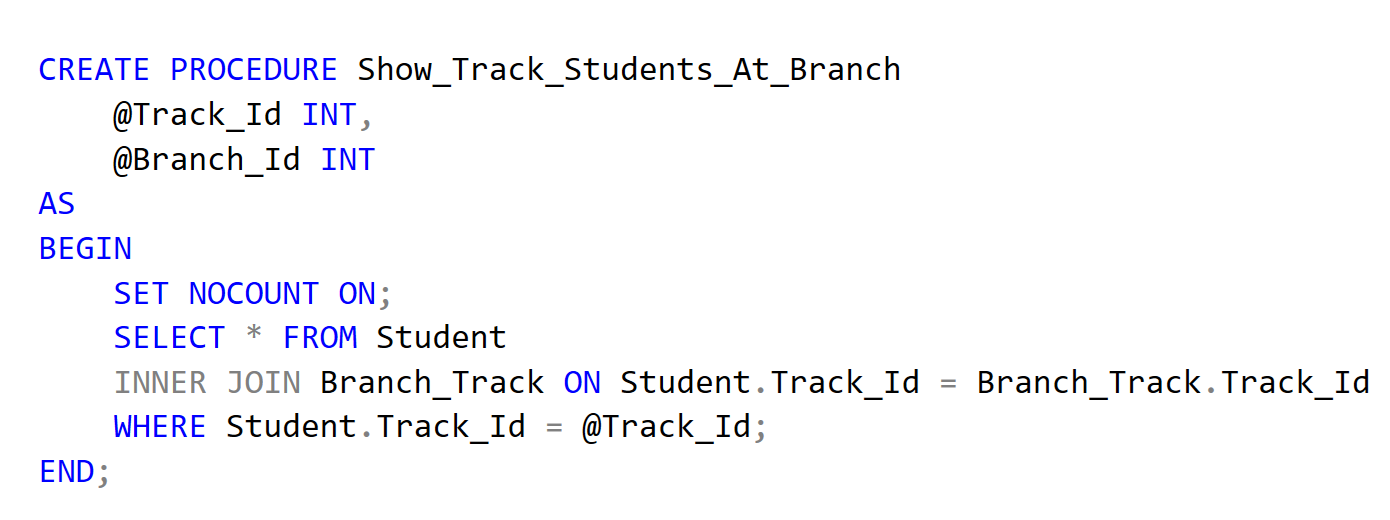
* 1. **Show\_Track\_Students**  
     **Description:**  
     This stored procedure retrieves all students enrolled in a specific track. It takes a Track\_Id as input and returns a list of all students who belong to that track.  
     **Parameters:**
* @Track\_Id (INT): The unique identifier of the track for which students will be retrieved.

**Output:**  
A list of students with details corresponding to the specified Track\_Id.



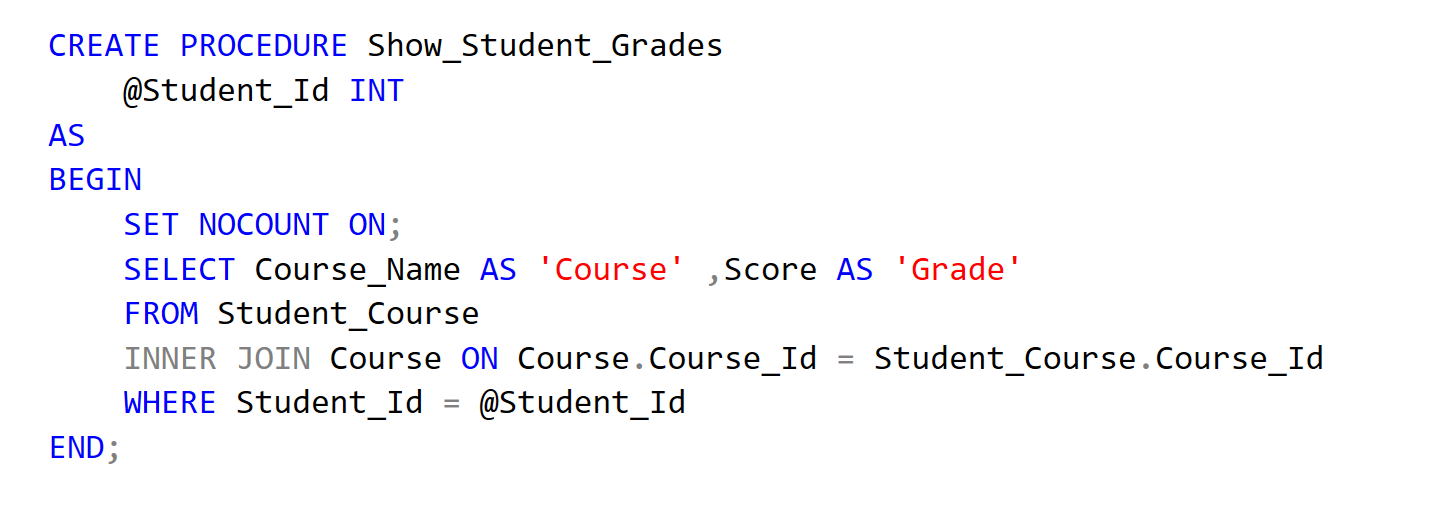
* 1. **Show\_Track\_Students\_At\_Branch**  
     **Description:**  
     This procedure retrieves students enrolled in a particular track and branch. It joins the Student table with the Branch\_Track table to filter students based on their track and branch.  
     **Parameters:**
* @Track\_Id (INT): The unique identifier for the track.
* @Branch\_Id (INT): The unique identifier for the branch.

**Output:**  
A list of students who belong to the specified track at the specified branch.



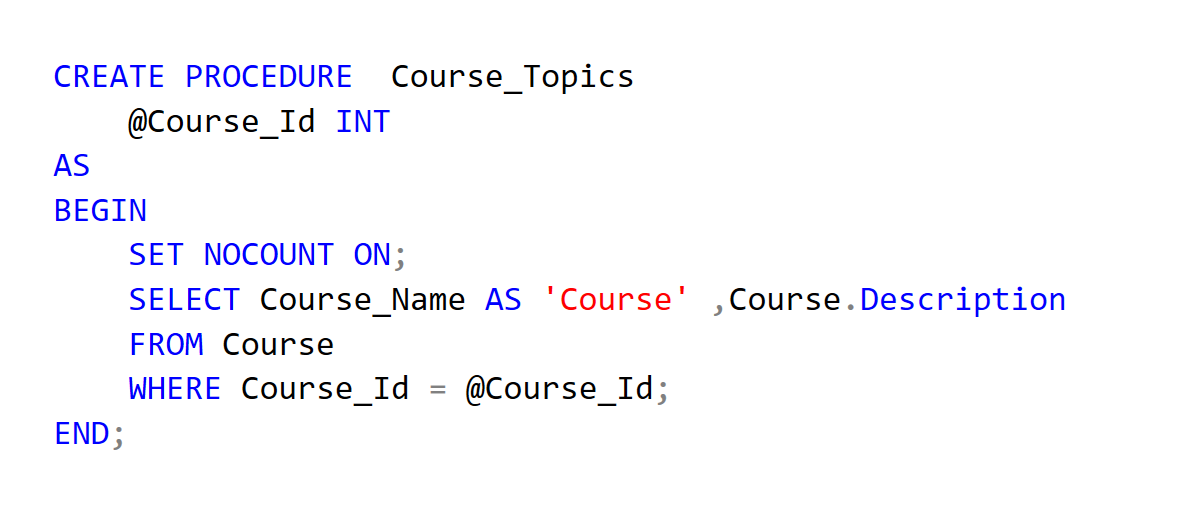
* 1. **Show\_Student\_Grades**  
     **Description:**  
     This stored procedure returns the courses and grades for a specific student. It uses the Student\_Course and Course tables to fetch the course names and corresponding scores for a given student.  
     **Parameters:**
* @Student\_Id (INT): The unique identifier of the student.

**Output:**  
A list of courses with their respective grades (score) for the specified student.



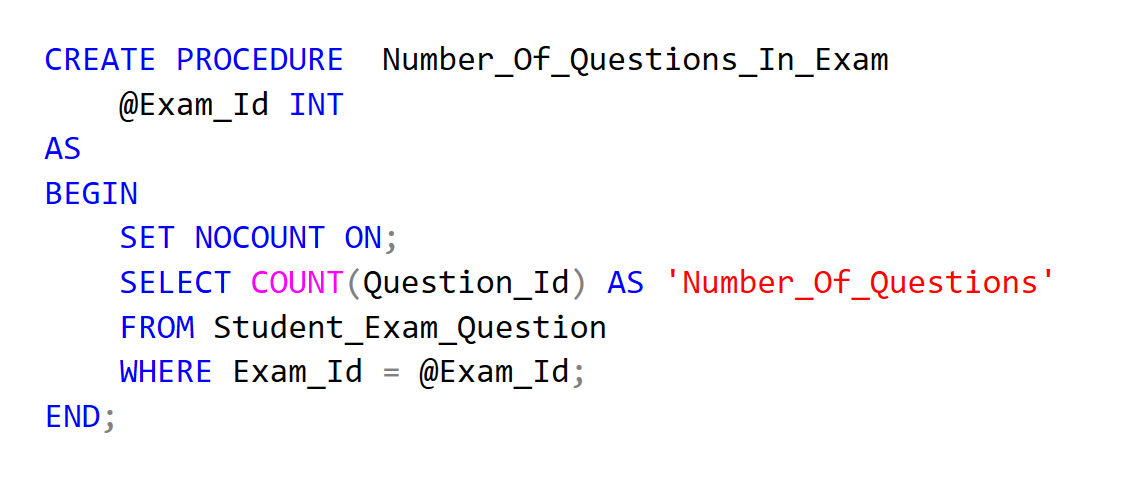
* 1. **Course\_Topics**  
     **Description:**  
     This procedure provides details about a specific course, including its name and description. It filters the Course table using the provided Course\_Id.  
     **Parameters:**
* @Course\_Id (INT): The unique identifier of the course for which details will be returned.

**Output:**  
The name and description of the specified course.



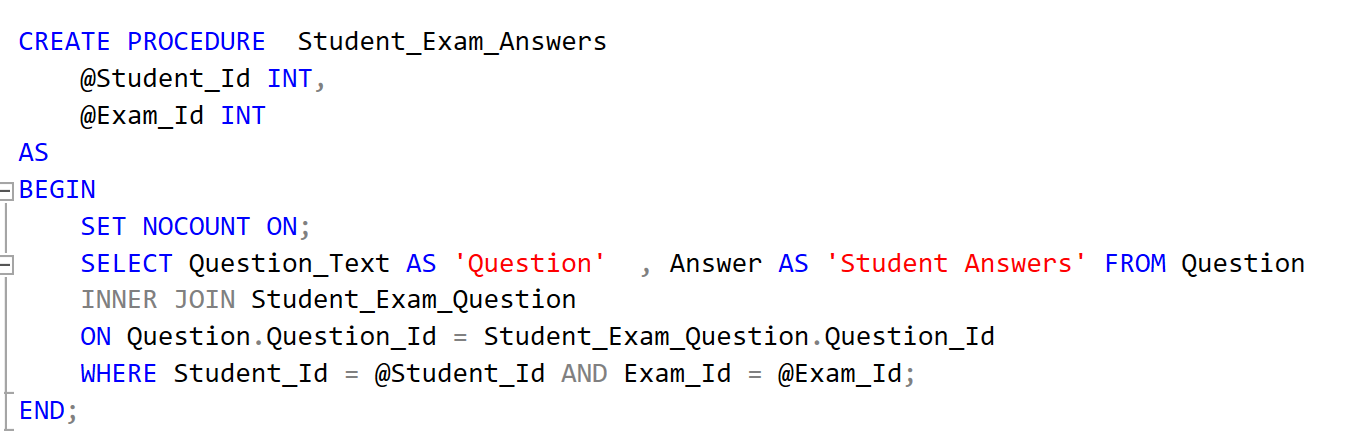
* 1. **Number\_Of\_Questions\_In\_Exam**  
     **Description:**  
     This stored procedure counts the total number of questions in a given exam by querying the Student\_Exam\_Question table. It returns the number of questions associated with the specified Exam\_Id.  
     **Parameters:**
* @Exam\_Id (INT): The unique identifier of the exam.

**Output:**  
A count of the number of questions associated with the specified exam.



* 1. **Student\_Exam\_Answers**  
     **Description:**  
     This procedure retrieves the questions and corresponding answers provided by a student during a specific exam. It joins the Question table with the Student\_Exam\_Question table to fetch the question text and the student's answer for the specified exam.  
     **Parameters:**
* @Student\_Id (INT): The unique identifier of the student.
* @Exam\_Id (INT): The unique identifier of the exam.

**Output:**  
A list of questions and the student's answers for the specified exam.



# **Data Base Backup:**

githubLink:

[**YousryEssam/ExaminationSystem: Database for Examination System**](https://github.com/YousryEssam/ExaminationSystem)

***Thank You!***