ITI Examination System

Database project

[GitHub]

Supervisor: Eng. Ramy

Presented by:

- Mohammed Fawzi
- Aya Mohamed Nafed
- Alaa Shokry
- Ashrakat Sami
- Gehad Shalaby

Table Of Contents

Abstract

The Examination System Database project outlines the development of a comprehensive SQL Server-based system designed to manage and automate the examination process for educational institutions. The system supports multiple question types, including multiple-choice, true/false, and text-based questions, with automated grading for objective questions and manual review for subjective answers. It facilitates course and user management, allowing instructors to create and schedule exams, assign students, and calculate results. A Training Manager role oversees the management of courses, instructors, branches, tracks, and student enrollments. The system enforces role-based access control, ensuring data security and integrity through SQL Server users and permissions. Technical requirements include optimized database design with appropriate data types, indexing, constraints, and triggers, along with the use of stored procedures, functions, and views for task automation. Daily automated backups are implemented to ensure data safety. Deliverables include system requirement documentation, an Entity-Relationship Diagram (ERD), database files, SQL scripts, test sheets, and a user accounts document. The system aims to provide a secure, efficient, and scalable platform for managing examinations, ensuring robust performance and streamlined workflows.

Introduction

The Examination System Database project is designed to create a robust and efficient system for managing examinations in an educational institution using SQL Server. This system aims to streamline the entire examination process, from question creation and exam scheduling to result calculation and reporting. Below is an explanation of the key processes involved in the system:

- **Question Pool Management:**

The system provides a centralized question pool where instructors can create and store questions of various types, including multiple-choice, true/false, and text-based questions. For objective questions (multiple-choice and true/false), the system automatically checks student answers against the correct answers stored in the database. For subjective (text-based) questions, the system stores the best-accepted answer and provides tools for instructors to manually review and grade student responses using text functions and regular expressions.

- **Course and User Management**:

The system maintains detailed information about courses, including course names, descriptions, maximum and minimum degree requirements. It also stores information about instructors and students, including their personal data, roles, and assigned courses. A Training Manager (a designated instructor) has elevated privileges to manage courses, instructors, branches, tracks, and student enrollments. This role ensures that the system is properly configured and maintained.

- **Exam Creation and Scheduling:**

Instructors can create exams by selecting questions from the question pool, either randomly or manually. Each exam is associated with a specific course, intake, branch, and track. Instructors can define the exam type (e.g., regular or corrective), start and end times, total duration, and allowance options. The system ensures that the total marks for the exam do not exceed the course's maximum degree

- **Student Exam Access and Answer Submission**:

Instructors can specify which students are eligible to take a particular exam. Students can only access and attempt the exam during the scheduled time. The system stores student answers and automatically grades objective questions. For subjective questions, instructors manually review and grade the responses.

- **Result Calculation and Reporting**:

The system calculates the results for each student based on their performance in the exam. Results are stored in the database, and instructors can generate reports to review student performance, including valid and invalid answers for subjective questions.

- **User Access and Security**:

The system supports four types of user accounts: Admin, Training Manager, Instructor, and Student. Each account type has specific permissions, ensuring that users can only access and perform tasks relevant to their role. SQL Server users and permissions are implemented to enforce data security and integrity

Implementation Life Cycle

The Implementation Life Cycle for the Examination System Database involves a structured process to design, develop, and deploy the system. Below is a concise explanation of each stage:

1. Requirements Gathering: This stage involves collecting and documenting all system requirements, such as question pool management, exam creation, and user roles (Admin, Training Manager, Instructor, Student).

2. Entity-Relationship Diagram (ERD) Design: The ERD is designed to visualize the database structure, including entities like Courses, Instructors, Students, and Exams, along with their relationships.

3. Mapping ERD to Database Schema: The ERD is converted into a physical database schema. Tables, columns, primary keys, foreign keys, and constraints are defined.

4. Online Schema Implementation: The schema is implemented in SQL Server using SQL Server Management Studio (SSMS) or scripts. Tables, relationships, and constraints are created

5. Data Insertion: Test data is inserted into tables like Courses, Instructors, and Students.

6. Database Operations[GitHub]: CRUD (Create, Read, Update, Delete) operations are performed on all tables. This ensures the database can handle basic operations like adding, retrieving, updating, and deleting records.

7. Stored Procedures: Stored procedures are created for optimized data retrieval and manipulation. They handle tasks like fetching exam results or updating student information, improving efficiency and security.

8. Triggers: Triggers are implemented to enforce data integrity and automate tasks. For example, a trigger can update student results when an exam is graded or prevent the deletion of a course with active exams.

9. Reports Using SSRS: Reports are generated using SQL Server Reporting Services (SSRS). These reports provide insights into exam results, student performance, and course statistics, helping stakeholders make informed decisions.

Exam Life Cycle

The Exam Life Cycle in the Examination System Database outlines the stages involved in creating, administering, and evaluating exams. Below is a concise explanation of each stage:

1. Question Creation:

Instructors create and store questions in the Question Pool, multiple-choice, true/false.

2.Exam Creation:

Instructors create exams by selecting questions from the pool, either randomly or manually.

3.Exam Scheduling:

Exams are scheduled with specific start and end times.

4. Exam Administration:

Students take the exam during the scheduled time, and their answers are recorded.

5. Answer Evaluation:

The system automatically grades objective questions.

6. Result Calculation:

The system calculates results based on student answers and stores them in the database.

7. Result Reporting:

Instructors and students can assess exam results. Reports are generated to analyze student performance.

Relationships

1. Branch – Track (M: N):

- Each Branch can have multiple Tracks.
- Each Track can belong to multiple Branches.

2. Instructor – Track (M: N):

- Each Instructor can teach in multiple Tracks.
- Each Track can have multiple Instructors
- This relationship includes an attribute for Hire Data

3. Instructor– Track (1: 1):

- Each Instructor is assigned to one Track for management purposes.
- Each Track has one Instructor responsible for its management.
- This relationship includes an attribute for Promotion Data

4. Track – Student (1: M):

- Each Track can have multiple Students.
- Each Student belongs to one Track.
- This relationship includes an attribute for Duration, start date and end date.

5. Track – Course (M: N):

- Each Track can offer multiple Courses.
- Each Course can belong to multiple Tracks.

6. Instructor – Course (1:M):

- Each Instructor can teach multiple Courses.
- Each Course is taught by one Instructor.

7. Topic – Course (M: 1):

- Each Topic can belong to one Course.
- Each Course can consist of multiple Topics.

8. Student– Course (M: N):

- Each Student can enroll in multiple Courses.
- Each Course can have multiple Students enrolled.

9. Student– Course - Exam (M: N):

- Each Student can attend multiple Exams for different Courses.
- Each Exam can be attended by multiple Students.
- This relationship includes an attribute for Std-Grade.

10. Course - Exam (1: M):

- Each Course can have multiple Exams.
- Each Exam belongs to one Course.

11. Question– Exam (M: N):

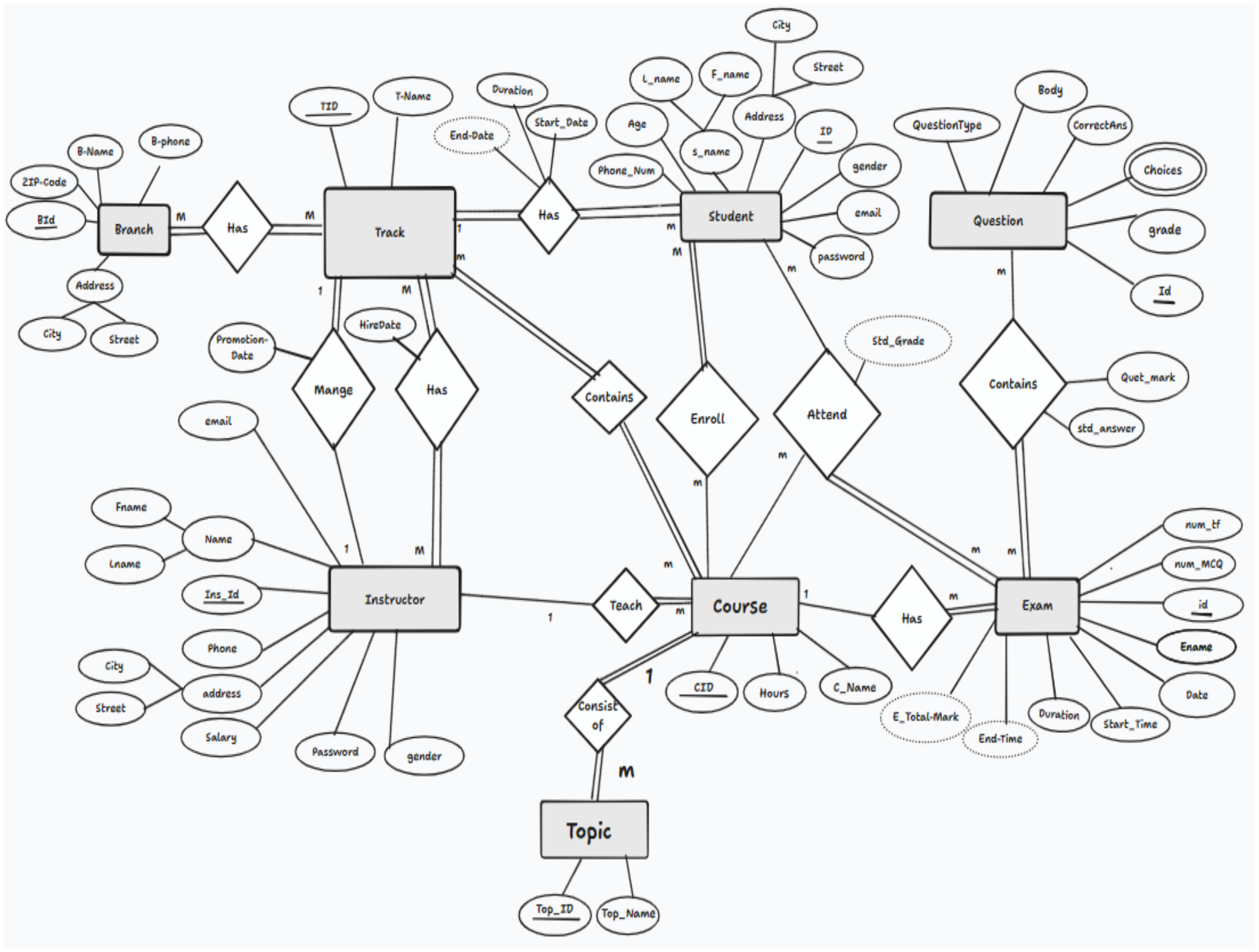
- Each Question can be included in multiple Exams.
- Each Exam can contain multiple Questions.
- This relationship includes an attribute for Quet_mark and std_answer.

Entity Relationship Diagram (ERD)

The ERD represents an examination management system for an educational institution. It includes the following key entities:

1. **Student**: Represents students enrolled in the system.
2. **Branch**: Represents different branches of the institution.
3. **Course**: Represents courses offered by the institution.
4. **Track**: Represents educational tracks or specializations.
5. **Instructor**: Represents instructors who teach courses.
6. **Exam**: Represents exams conducted for courses.
7. **Question:** Represents a pool of questions for exams.
8. **Topic:** List of Topics Included in a Course

ERD


Branch
Has
Track
Has
Student
Question
BId
ZIP-Code
B-Name
B-phone
Address
City
Street
TID
T-Name
Duration
End-Date
Start_Date
L_name
F_name
City
Street
Address
Age
S_name
ID
Phone_Num
gender
email
password
QuestionType
Body
CorrectAns
Choices
grade
Id
Promotion-Date
Mange
HireDate
Has
Contains
Enroll
Attend
Std_Grade
Contains
Quet_mark
std_answer
email
Fname
Name
Lname
Ins_Id
Instructor
Phone
City
address
Street
Salary
Password
gender
Teach
Course
CID
Hours
C_Name
Has
Exam
num_tf
num_MCQ
id
Ename
Date
Start_Time
Duration
End-Time
E_Total-Mark
Consist of
Topic
Top_ID
Top_Name
M
M
1
m
m
M
1
M
1
M
m
m
m
m
m
m
m
1
m
1
m
m
1
m

Tables

1-Instructor Table

➢ **Description:** This table stores information about instructors, including their personal details, contact information, and employment data.

Column Name	Data Type	Description	Constraints
Ins_id	INT	Unique identifier for the instructor	Primary Key
Ins_Fname	VARCHAR (100)	First name of the instructor	NOT NULL
Ins_Lname	VARCHAR (100)	Last name of the instructor	NOT NULL
City	VARCHAR (100)	City where the instructor resides	NULLABLE
Street	VARCHAR (255)	Street address of the instructor	NULLABLE
Phone	VARCHAR (20)	Phone number of the instructor	NULLABLE
Email	VARCHAR (255)	Email address of the instructor	NULLABLE
Password	VARCHAR (255)	Login password for the instructor	NULLABLE
Salary	DECIMAL (10,2)	Salary of the instructor	NULLABLE
Promotion_Date	DATE	Date of the last promotion	NULLABLE
Gender	CHAR (1)	Gender of the instructor (M/F)	NULLABLE

2-Track_Course_Contain Table

- **Description:** This table manages the many-to-many relationship between tracks and courses.

Column Name	Data Type	Description	Constraints
`C_id`	INT	Unique identifier for the course	Primary Key, Foreign Key referencing `Course(C_id)`
`T_id`	INT	Unique identifier for the track	Primary Key, Foreign Key referencing `Track(T_id)`

3- Ins_Track Table

- **Description:** This table manages the many-to-many relationship between instructors and tracks, including the hire date of the instructor for a specific track.

Column Name	Data Type	Description	Constraints
`Ins_id`	INT	Unique identifier for the instructor	Primary Key, Foreign Key referencing `Instructor (Ins_id)`
`T_id`	INT	Unique identifier for the track	Primary Key, Foreign Key referencing `Track(T_id)`
`HireDate`	DATE	The date the instructor was assigned to the track	NULLABLE

4- Branch Table

➢ **Description:** This table stores details about different branches of an organization, including their location, phone numbers, and manager information.

Column Name	Data Type	Description	Constraints
B_id	INT	Unique identifier for the branch	Primary Key
B_name	VARCHAR (255)	Name of the branch	NOT NULL
City	VARCHAR (100)	City where the branch is located	NOT NULL
ZIP_Code	VARCHAR (20)	ZIP code of the branch's location	NULLABLE
Street	VARCHAR (255)	Street address of the branch	NULLABLE
B_Phone	VARCHAR (20)	Phone number of the branch	NULLABLE
Mgr_Name	VARCHAR (20)	Name of the branch manager	NULLABLE
Mgr_Phone	VARCHAR (20)	Phone number of the branch manager	NULLABLE
Mgr_Selected_Date	DATE	Date when the manager was selected	NULLABLE

5- BranchTrack Table

➢ **Description:** This table manages the many-to-many relationship between branches and tracks.

Column Name	Data Type	Description	Constraints
B_id	INT	Unique identifier for the branch	Primary Key, Foreign Key referencing `Branch(B_id)`
T_id	INT	Unique identifier for the track	Primary Key, Foreign Key referencing `Track(T_id)`

6- Track Table

- **Description:** This table stores information about educational or organizational tracks, including the associated instructor.

Column	Data Type	Description	Constraints
T_id	INT	Unique identifier for the track	Primary Key
T_Name	INT	Name of the track	NOT NULL
FK_Ins_Id	VARCHAR (255)	Identifier for the instructor responsible for the track	Foreign Key referencing Instructor (Ins_id)

7- Exam Table

8- **Description:** This table stores details about exams conducted in the system. It includes metadata like exam date, duration, start time, and total marks.

Attribute	Type	Description	Constraints
Ex_Id	INT	Unique identifier for the exam.	Primary Key, Not Null
EName	VARCHAR (255)	Name of the exam.	Not Null
No_MCQ	INT	Number of MCQ questions.	Nullable
No_TF	INT	Number of True/False questions.	Nullable
Fk_CID	INT	Foreign Key referencing Course(C_id).	FOREIGN KEY (Fk_CID) REFERENCES Course(C_id)
Start_Time	DATETIME	Start time of the exam.	Nullable
Duration	INT	Duration of the exam.	Nullable

9- Std_Exam Table

➢ **Description:** This table stores details about student exams. It links students to specific exams, recording their grade, date, and duration.

Attribute	Type	Description	Constraints
SID	INT	Student ID.	Part of primary key, FOREIGN KEY (SID) REFERENCES Student(S_id)
Ex_ID	INT	Exam ID.	Part of primary key, FOREIGN KEY (Ex_ID) REFERENCES Exam (Ex_ID)
Grade	VARCHAR (10)	Grade obtained by the student.	Nullable
Date	DATE	Date of the exam.	Nullable
Duration	INT	Duration the student took for the exam.	Nullable

10- Exam_Attendance Table

➢ **Description:** This table tracks student attendance for each exam. It links students and courses to specific exams.

Attribute	Type	Description	Constraints
S_id	INT	Student ID.	Part of primary key, FOREIGN KEY (S_id) REFERENCES Student (S_id
C_id	INT	Course ID.	Part of primary key, FOREIGN KEY (C_id) REFERENCES Course(C_id)
Ex_id	INT	Exam ID.	Part of primary key, FOREIGN KEY (Ex_id) REFERENCES Exam (Ex_ID)

11- Topic Table

➢ **Description:** This table stores information about various topics. Each topic has a unique identifier and a name.

Attribute	Type	Description	Constraints
Top_id	INT	Unique identifier for each topic.	PRIMARY KEY
Top_name	VARCHAR (20)	Name of the topic, limited to 20 characters.	None

12- ExamQuestion Table

➢ **Description:** This table links exams to their questions and tracks the students' answers for those questions. It establishes a many-to-many relationship between the Exam and Question tables.

Column Name	Data Type	Description	Constraints
Ex_Id	INT	Foreign key referencing the exam to which the question belongs.	REFERENCES Exam (Ex_Id)
Q_id	INT	Foreign key referencing the unique identifier of the question.	REFERENCES Question(Q_id)
Std_Answer	VARCHAR (255)	The student's submitted answer for the specific question in the exam.	NULLABLE

13- Question Table

➢ **Description:** This table stores information about all the questions in the system, including their text, type, and grade value.

Column	Data Type	Description	Constraints
Q_id	INT	Primary key, a unique identifier for each question.	PRIMARY KEY
Header	VARCHAR (255)	A brief title or header for the question.	NOT NULL
Body	TEXT	The detailed text or body of the question.	NULLABLE
CorrectAns	VARCHAR (255)	The correct answer to the question, used for evaluation.	NULLABLE
Type	VARCHAR (50)	Indicates the type of question (e.g., MCQ, True/False, etc.).	NULLABLE
Grade	DECIMAL (5, 2)	The score/grade value assigned to the question.	NULLABLE

14- Choices Table

➢ **Description:** This table stores the multiple-choice options for questions. It supports a many-to-one relationship, where a question can have multiple choices.

Column	Data Type	Description	Constraints
Q_id	INT	Foreign key referencing the unique identifier of the question.	REFERENCES Question(Q_id), NOT NULL
Choice_Id	INT	Unique identifier for each choice within a question (composite key with Q_id).	PRIMARY KEY (Q_id, Choice_Id)
Choice_Text	VARCHAR (255)	The text of the choice option.	NULLABLE

14. Enroll_Crs_Std Table

➢ **Description**: This table links students to courses, with foreign keys referencing both the Course and Student tables.

Column Name	Data Type	Description	Constraints
C_id	INT	Unique identifier for the course	Primary Key, Foreign Key
S_id	INT	Unique identifier for the student	Primary Key, Foreign Key

15. Student Table

➢ **Description**: This table stores information about students, including their personal details, contact information, and academic details.

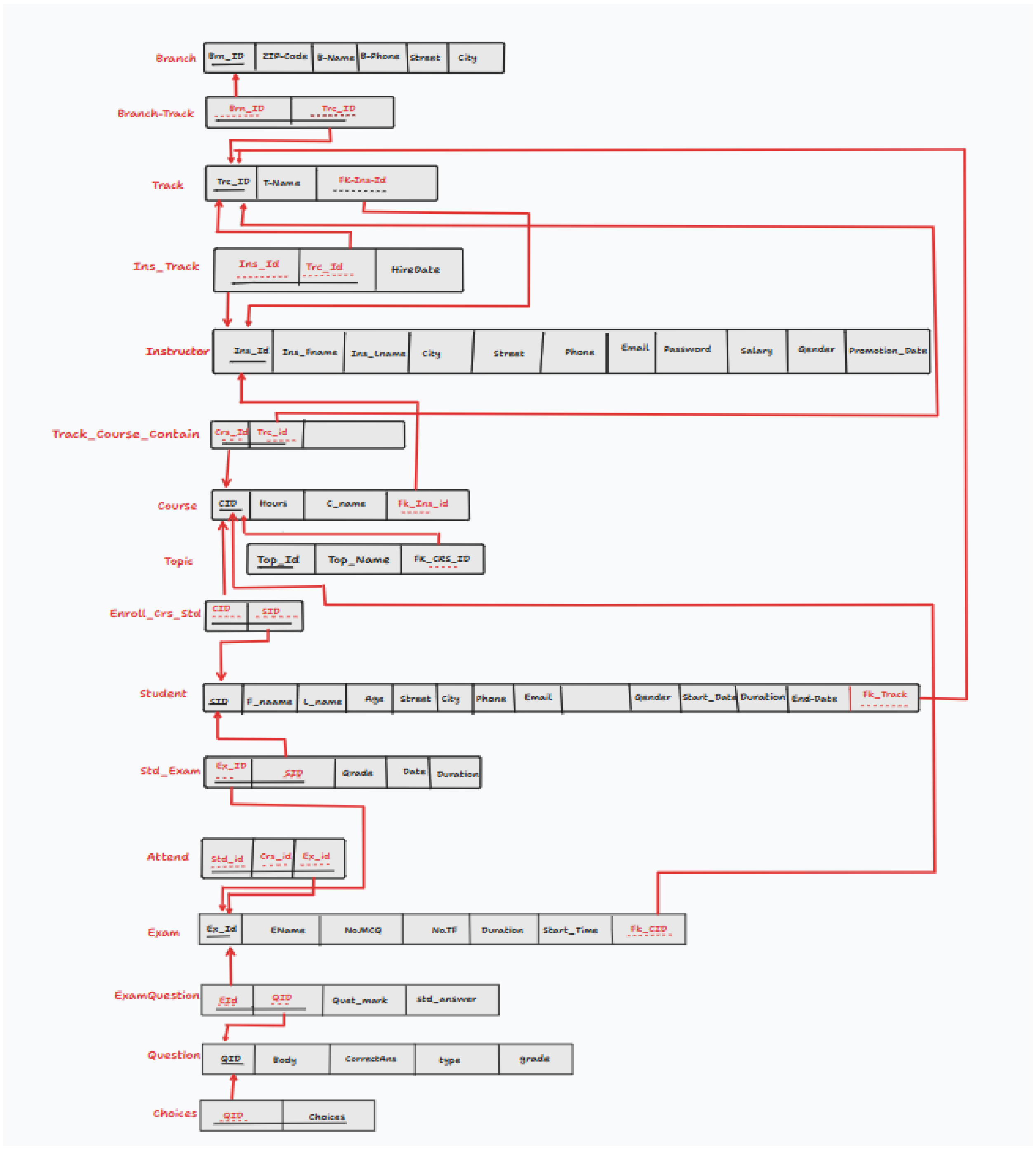
Column Name	Data Type	Description	Constraints
S_id	INT	Unique identifier for the student	Primary Key
F_Name	VARCHAR (100)	First name of the student	NOT NULL
L_Name	VARCHAR (100)	Last name of the student	NOT NULL
Phone	VARCHAR (20)	Phone number of the student	NULLABLE
Email	VARCHAR (255)	Email address of the student	NULLABLE
Password	VARCHAR (255)	Login password for the student	NULLABLE
Gender	CHAR (1)	Gender of the student (M/F)	NULLABLE
Age	INT	Age of the student	NULLABLE
Fk_Track	INT	Foreign key to the Track table	NULLABLE
Street	VARCHAR (255)	Street address of the student	NULLABLE
City	VARCHAR (100)	City where the student resides	NULLABLE
Start_Date	DATE	Start date of the course	NULLABLE
End_Date	DATE	End date of the course	NULLABLE

16.Course Table

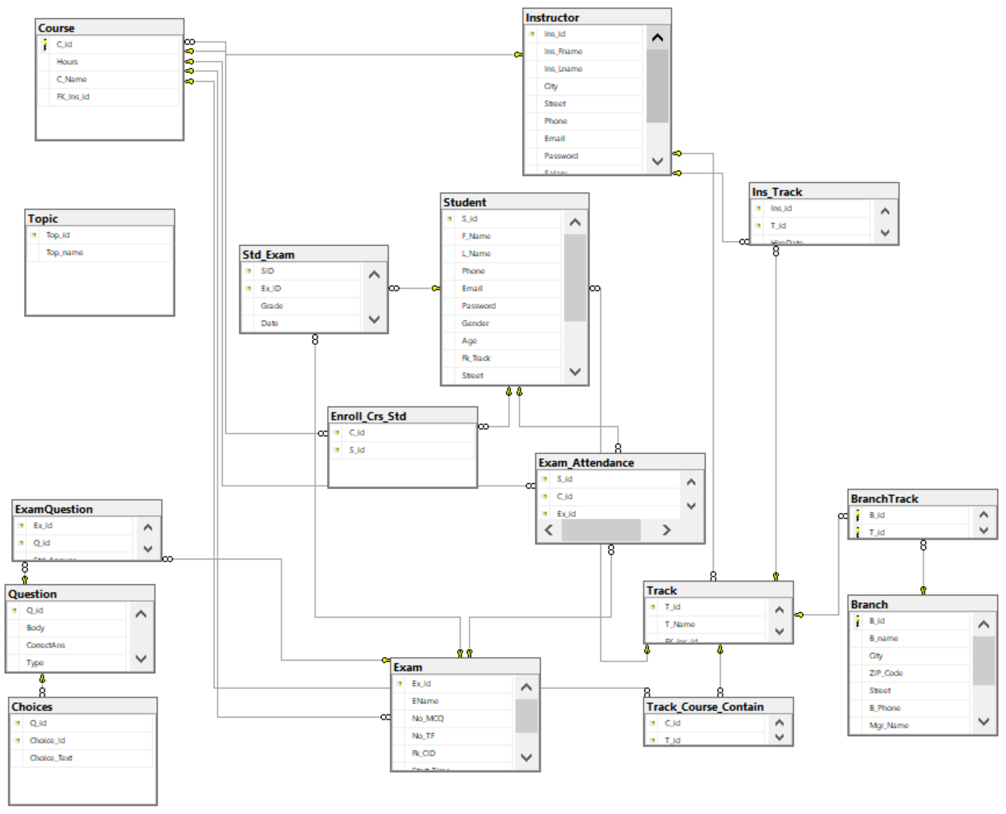
- **Description**: This table stores information about courses, including their details and the instructors teaching them.

Column Name	Data Type	Description	Constraints
C_id	INT	Unique identifier for the course	Primary Key
Hours	INT	Number of hours of the course	NOT NULL
C_Name	VARCHAR (255)	Name of the course	NOT NULL
FK_Ins_id	INT	Foreign key to the instructor table	NULLABLE

Mapping

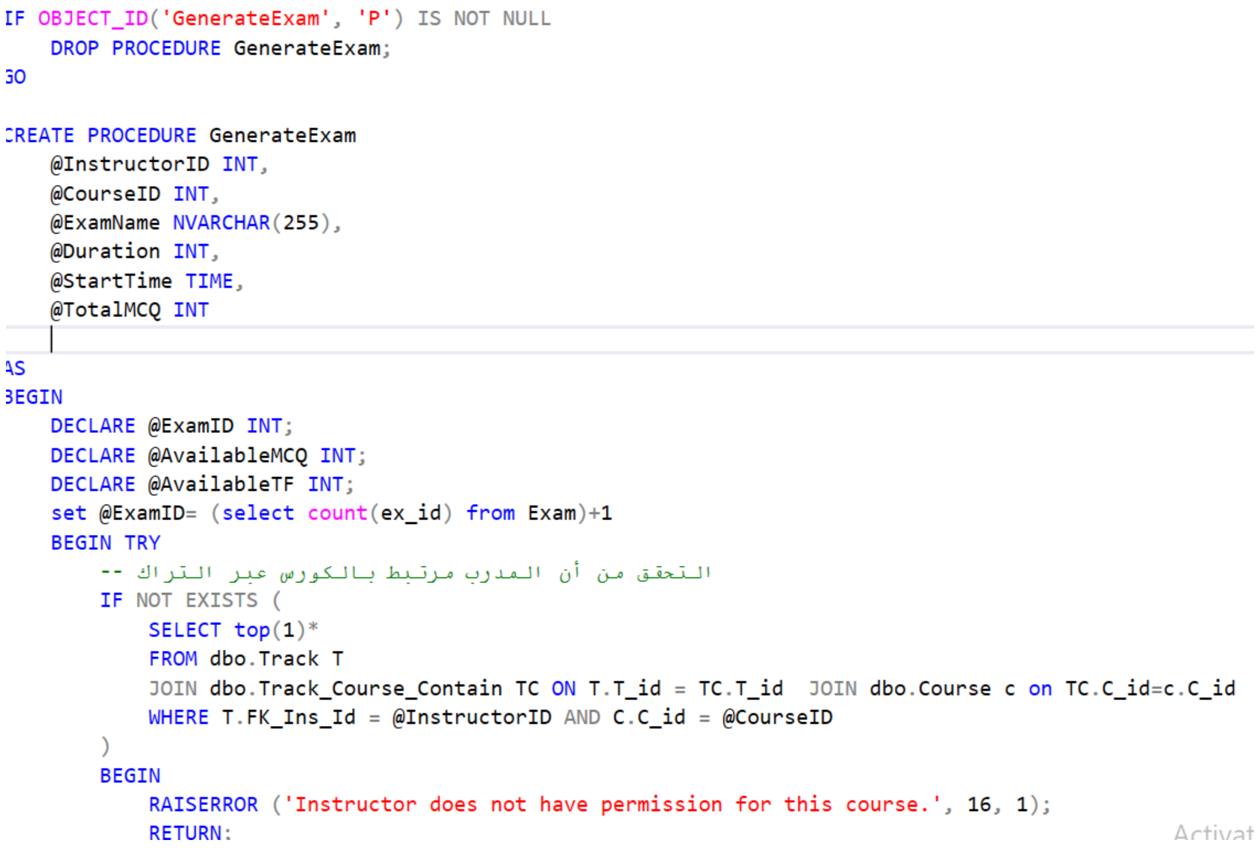

Branch
Brn_ID
ZIP-Code
B-Name
B-Phone
Street
City
Branch-Track
Brn_ID
Trc_ID
Track
Trc_ID
T-Name
FK-Ins-Id
Ins_Track
Ins_Id
Trc_Id
HireDate
Instructor
Ins_Id
Ins_fname
Ins_Lname
City
Street
Phone
Email
Password
Salary
Gender
Promotion_Date
Track_Course_Contain
Crs_Id
Trc_id
Course
CID
Hours
C_name
Fk_Ins_id
Topic
Top_Id
Top_Name
FK_CRS_ID
Enroll_Crs_Std
CID
SID
Student
SID
F_naame
L_name
Age
Street
City
Phone
Email
Gender
Start_Date
Duration
End-Date
Fk_Track
Std_Exam
Ex_ID
SID
Grade
Date
Duration
Attend
Std_id
Crs_id
Ex_id
Exam
Ex_Id
EName
No.MCQ
No.TF
Duration
Start_Time
Fk_CID
ExamQuestion
EId
QID
Qust_mark
std_answer
Question
QID
Body
CorrectAns
type
grade
Choices
QID
Choices

Database Diagram



Stored Procedure: ExamGeneration_sp

The Exam Generation process is handled by a stored procedure that automates the creation of exams.

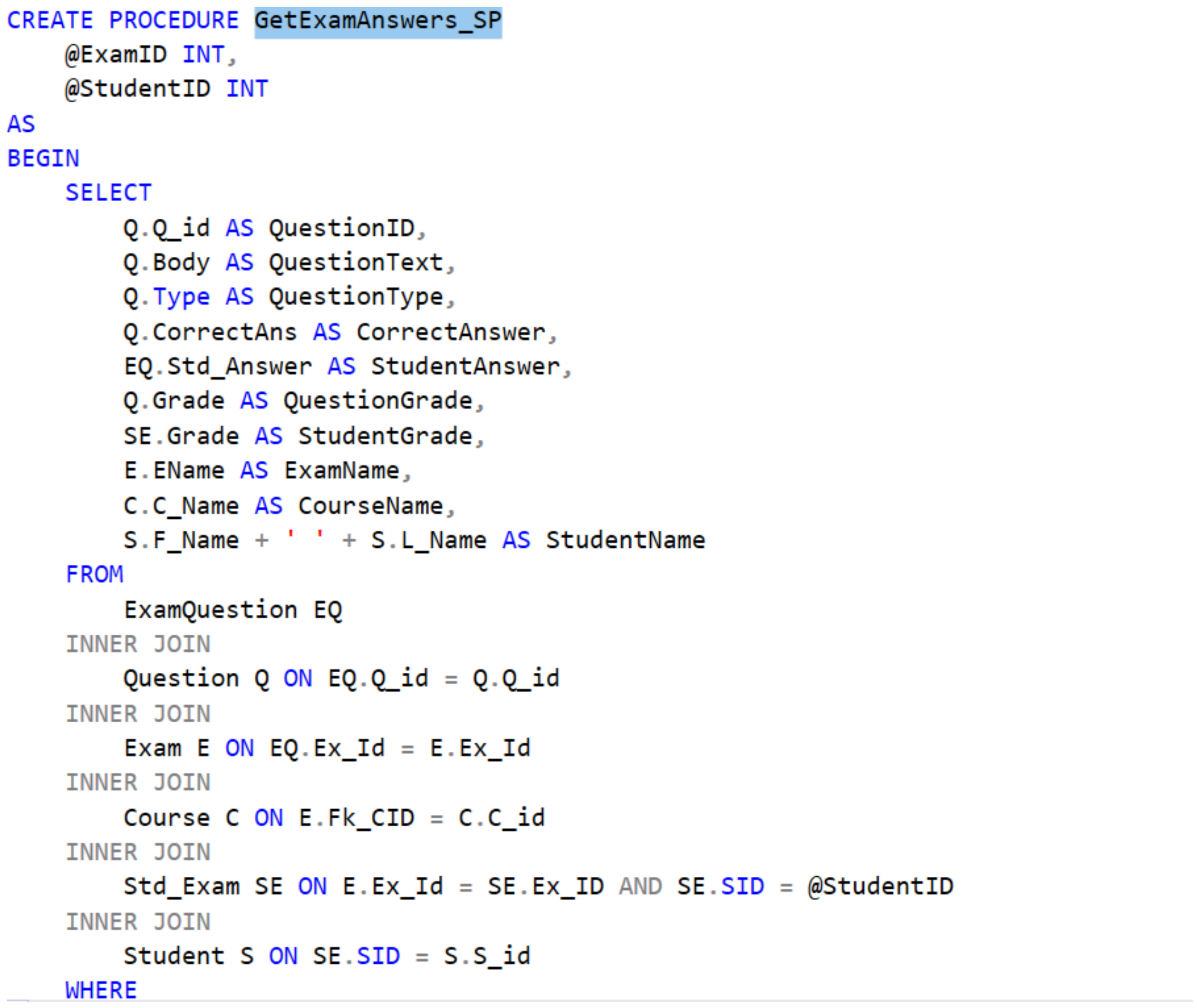


```
IF OBJECT_ID('GenerateExam', 'P') IS NOT NULL
    DROP PROCEDURE GenerateExam;
3O

:REATE PROCEDURE GenerateExam
    @InstructorID INT,
    @CourseID INT,
    @ExamName NVARCHAR(255),
    @Duration INT,
    @StartTime TIME,
    @TotalMCQ INT

AS
3EGIN
    DECLARE @ExamID INT;
    DECLARE @AvailableMCQ INT;
    DECLARE @AvailableTF INT;
    set @ExamID= (select count(ex_id) from Exam)+1
    BEGIN TRY
        -- التحقق من أن المدرب مرتبط بالكورس عبر التراك
        IF NOT EXISTS (
            SELECT top(1)*
            FROM dbo.Track T
            JOIN dbo.Track_Course_Contain TC ON T.T_id = TC.T_id  JOIN dbo.Course c on TC.C_id=c.C_id
            WHERE T.FK_Ins_Id = @InstructorID AND C.C_id = @CourseID
        )
        BEGIN
            RAISERROR ('Instructor does not have permission for this course.', 16, 1);
            RETURN;
```

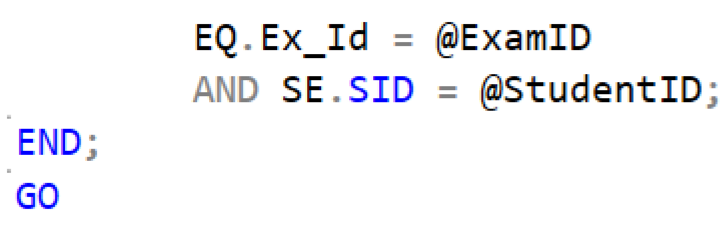
```sql
END
-- التحقق من وجود عدد كافٍ من الأسئلة
SELECT @AvailableMCQ = COUNT(*)
FROM dbo.Question q join ExamQuestion eq on q.Q_id=eq.Q_id
join exam e on eq.Ex_Id=e.Ex_Id join Course c on e.Fk_CID=c.C_id
WHERE e.Fk_CID = @CourseID AND q.Type = 'Choice';

SELECT @AvailableTF = COUNT(*)
 FROM dbo.Question q join ExamQuestion eq on q.Q_id=eq.Q_id
join exam e on eq.Ex_Id=e.Ex_Id join Course c on e.Fk_CID=c.C_id
WHERE Fk_CID = @CourseID AND Type = 'True/False';

IF @AvailableMCQ < @TotalMCQ
BEGIN
    RAISERROR ('Not enough questions available for the specified exam.', 16, 1);
    RETURN;
END
-- إدخال بيانات الامتحان في جدول Exam
INSERT INTO dbo.Exam (Ex_Id,EName, No_MCQ, No_TF, Fk_CID, Start_Time, Duration)
VALUES (@ExamID,@ExamName, @TotalMCQ,2-@TotalMCQ, @CourseID, @StartTime, @Duration);



INSERT INTO dbo.ExamQuestion (Ex_Id, Q_id)
SELECT TOP (@TotalMCQ) @ExamID, q.Q_id
FROM dbo.Question q join ExamQuestion eq on q.Q_id=eq.Q_id
join exam e on eq.Ex_Id=e.Ex_Id join Course c on e.Fk_CID=c.C_id
WHERE Fk_CID = @CourseID AND Type = 'Choice'
ORDER BY NEWID();



INSERT INTO dbo.ExamQuestion (Ex_Id, Q_id)
SELECT TOP (2-@TotalMCQ) @ExamID, q.Q_id
FROM dbo.Question q join ExamQuestion eq on q.Q_id=eq.Q_id
join exam e on eq.Ex_Id=e.Ex_Id join Course c on e.Fk_CID=c.C_id
WHERE Fk_CID = @CourseID AND Type = 'True/False'
ORDER BY NEWID();

PRINT 'Exam created success'

END TRY
    BEGIN CATCH
        PRINT 'An error occurred while creating the exam.';
        THROW;
    END CATCH
END;
```

Stored Procedure: GetExamAnswers_SP

The GetExamAnswers stored procedure retrieves student answers for a specific exam



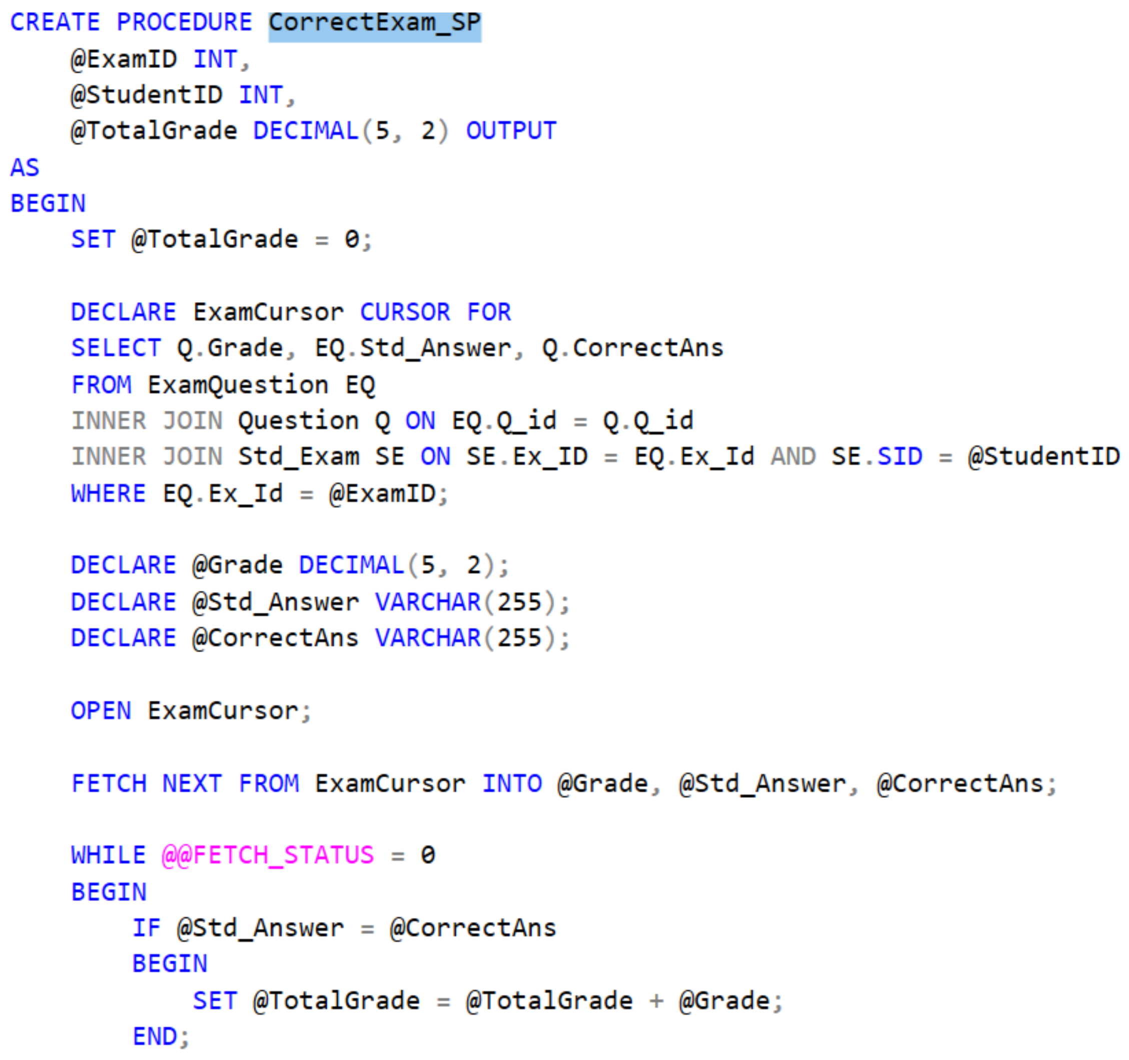
```sql
CREATE PROCEDURE GetExamAnswers_SP
    @ExamID INT,
    @StudentID INT
AS
BEGIN
    SELECT
        Q.Q_id AS QuestionID,
        Q.Body AS QuestionText,
        Q.Type AS QuestionType,
        Q.CorrectAns AS CorrectAnswer,
        EQ.Std_Answer AS StudentAnswer,
        Q.Grade AS QuestionGrade,
        SE.Grade AS StudentGrade,
        E.EName AS ExamName,
        C.C_Name AS CourseName,
        S.F_Name + ' ' + S.L_Name AS StudentName
    FROM
        ExamQuestion EQ
    INNER JOIN
        Question Q ON EQ.Q_id = Q.Q_id
    INNER JOIN
        Exam E ON EQ.Ex_Id = E.Ex_Id
    INNER JOIN
        Course C ON E.Fk_CID = C.C_id
    INNER JOIN
        Std_Exam SE ON E.Ex_Id = SE.Ex_ID AND SE.SID = @StudentID
    INNER JOIN
        Student S ON SE.SID = S.S_id
    WHERE
```



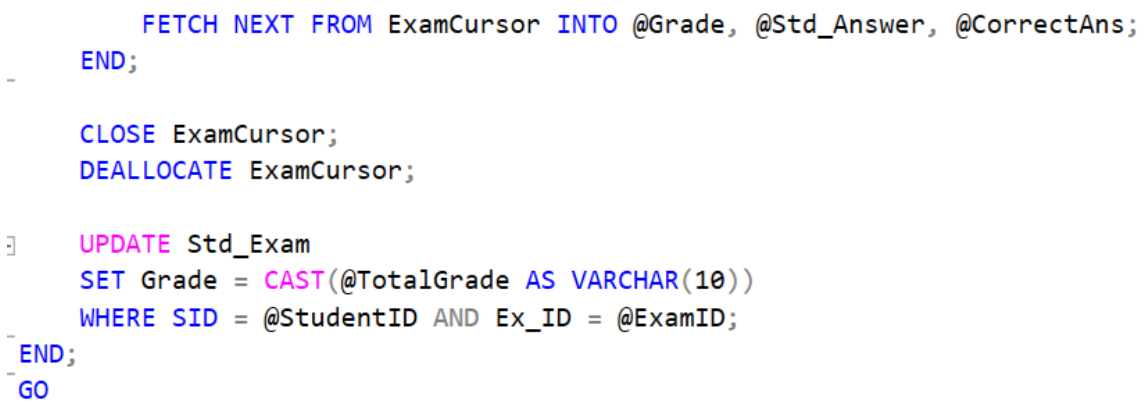
```sql
        EQ.Ex_Id = @ExamID
        AND SE.SID = @StudentID;
END;
GO
```

Stored Procedure: CorrectExam_SP

The CorrectExam_SP stored procedure automates the grading process for exams.

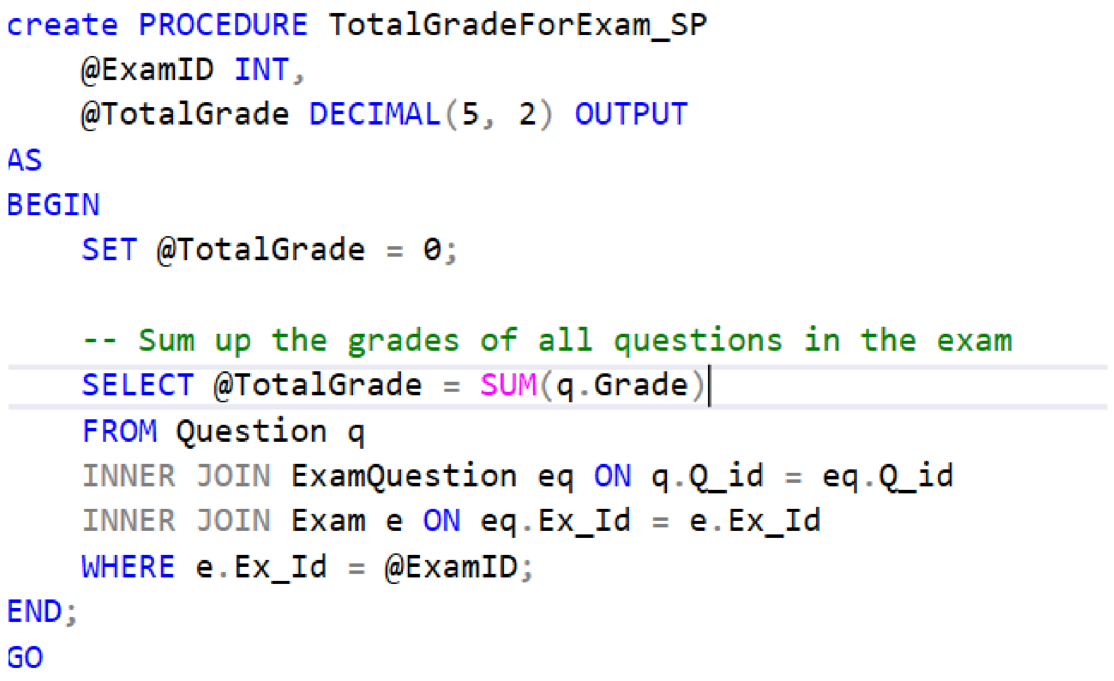


```
CREATE PROCEDURE CorrectExam_SP
    @ExamID INT,
    @StudentID INT,
    @TotalGrade DECIMAL(5, 2) OUTPUT
AS
BEGIN
    SET @TotalGrade = 0;

    DECLARE ExamCursor CURSOR FOR
    SELECT Q.Grade, EQ.Std_Answer, Q.CorrectAns
    FROM ExamQuestion EQ
    INNER JOIN Question Q ON EQ.Q_id = Q.Q_id
    INNER JOIN Std_Exam SE ON SE.Ex_ID = EQ.Ex_Id AND SE.SID = @StudentID
    WHERE EQ.Ex_Id = @ExamID;

    DECLARE @Grade DECIMAL(5, 2);
    DECLARE @Std_Answer VARCHAR(255);
    DECLARE @CorrectAns VARCHAR(255);

    OPEN ExamCursor;

    FETCH NEXT FROM ExamCursor INTO @Grade, @Std_Answer, @CorrectAns;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        IF @Std_Answer = @CorrectAns
        BEGIN
            SET @TotalGrade = @TotalGrade + @Grade;
        END;
```



```
        FETCH NEXT FROM ExamCursor INTO @Grade, @Std_Answer, @CorrectAns;
    END;

    CLOSE ExamCursor;
    DEALLOCATE ExamCursor;

    UPDATE Std_Exam
    SET Grade = CAST(@TotalGrade AS VARCHAR(10))
    WHERE SID = @StudentID AND Ex_ID = @ExamID;
END;
GO
```

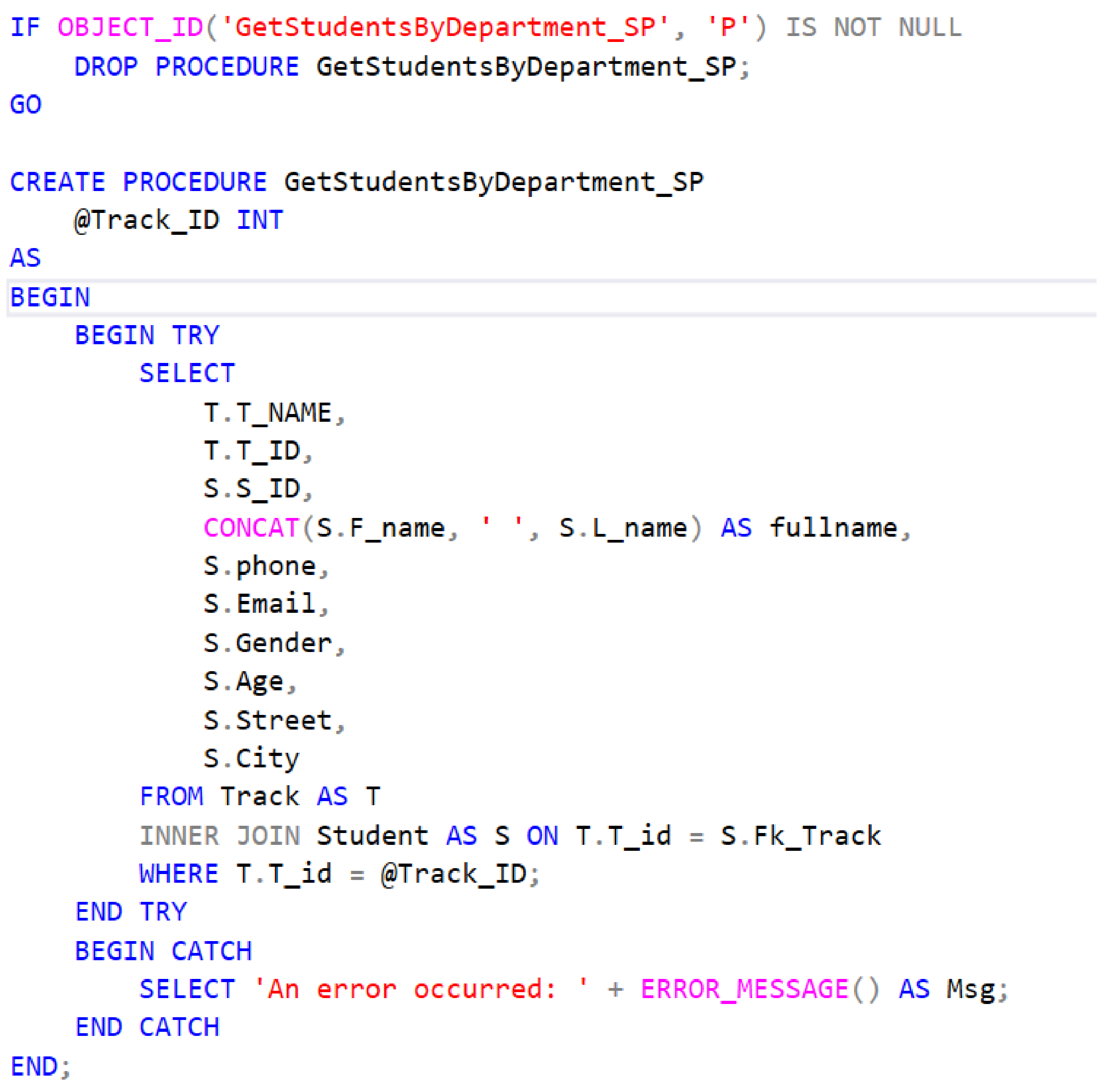
Stored Procedure: TotalGradeForExam_SP



```
create PROCEDURE TotalGradeForExam_SP
    @ExamID INT,
    @TotalGrade DECIMAL(5, 2) OUTPUT
AS
BEGIN
    SET @TotalGrade = 0;

    -- Sum up the grades of all questions in the exam
    SELECT @TotalGrade = SUM(q.Grade)
    FROM Question q
    INNER JOIN ExamQuestion eq ON q.Q_id = eq.Q_id
    INNER JOIN Exam e ON eq.Ex_Id = e.Ex_Id
    WHERE e.Ex_Id = @ExamID;
END;
GO
```

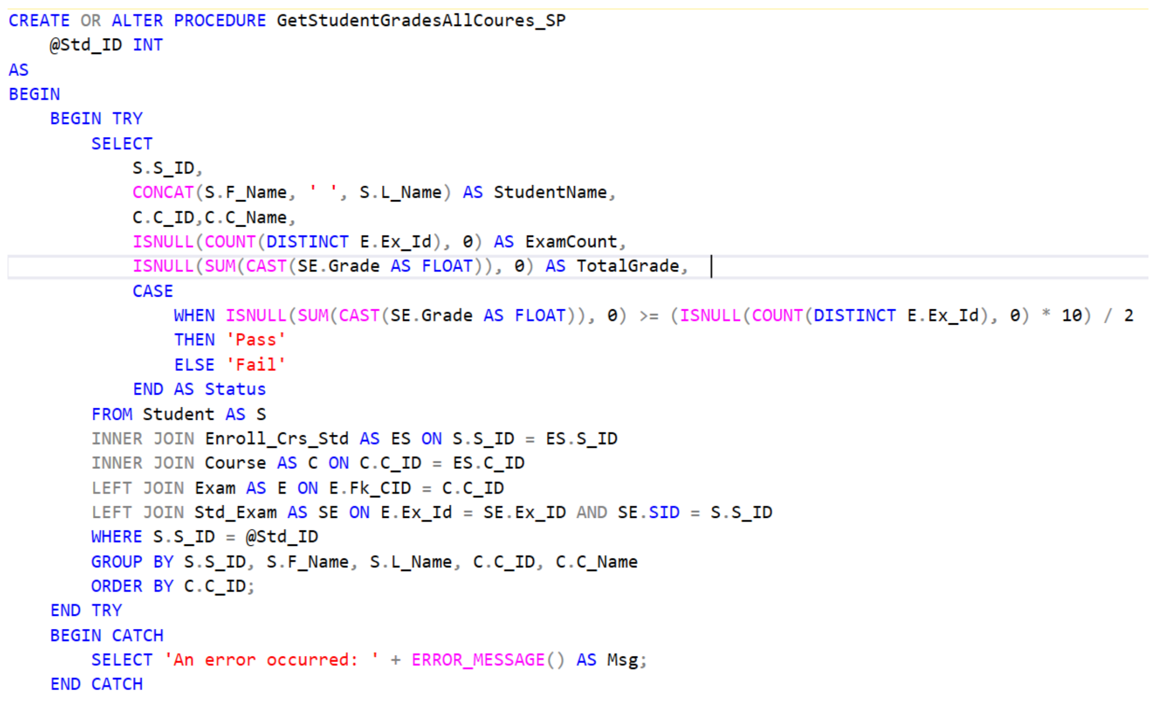
Stored Procedure for Report Generation

➢ Report that returns the students' information according to Department No parameter.



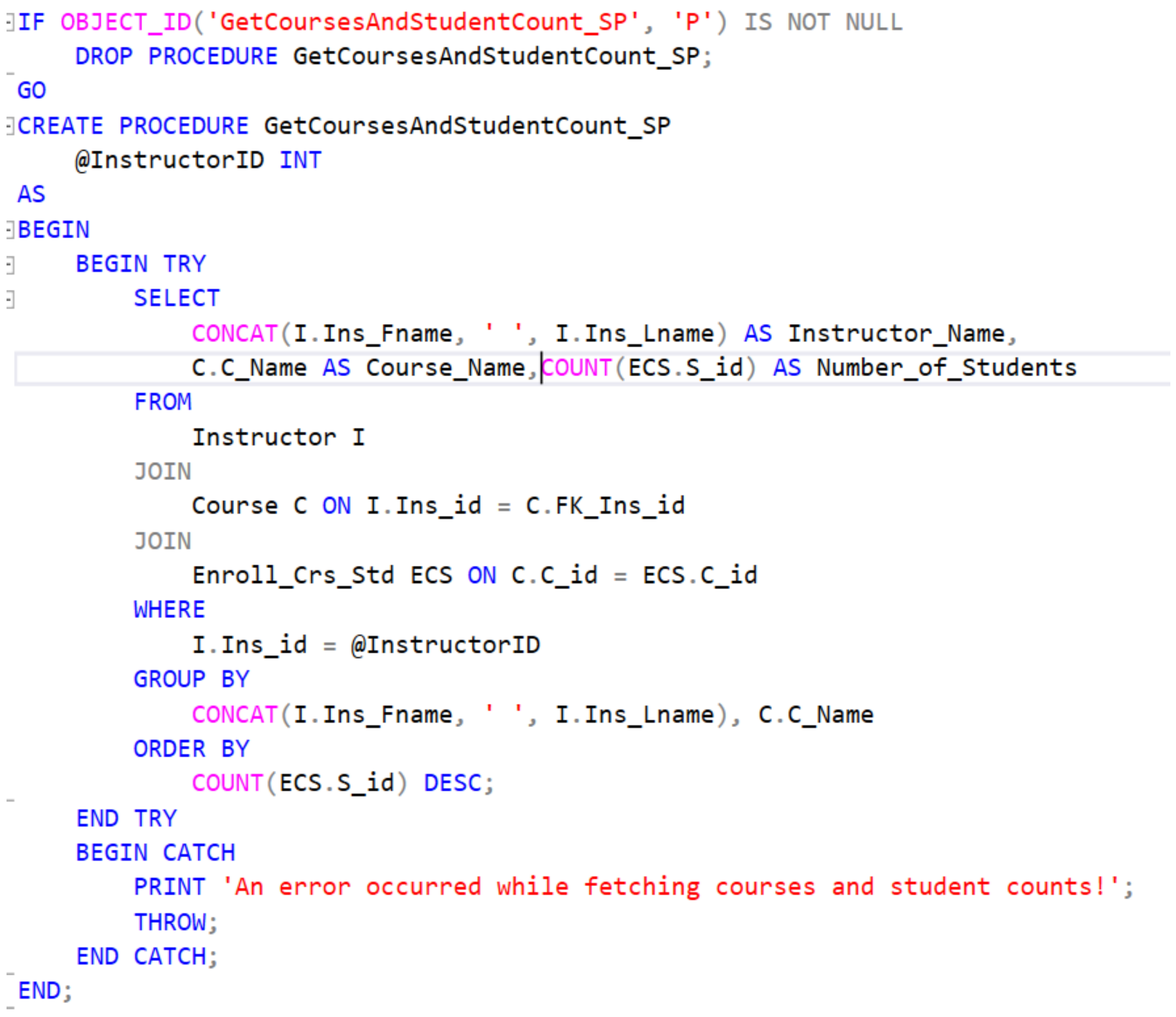
```sql
IF OBJECT_ID('GetStudentsByDepartment_SP', 'P') IS NOT NULL
    DROP PROCEDURE GetStudentsByDepartment_SP;
GO

CREATE PROCEDURE GetStudentsByDepartment_SP
    @Track_ID INT
AS
BEGIN
    BEGIN TRY
        SELECT
            T.T_NAME,
            T.T_ID,
            S.S_ID,
            CONCAT(S.F_name, ' ', S.L_name) AS fullname,
            S.phone,
            S.Email,
            S.Gender,
            S.Age,
            S.Street,
            S.City
        FROM Track AS T
        INNER JOIN Student AS S ON T.T_id = S.Fk_Track
        WHERE T.T_id = @Track_ID;
    END TRY
    BEGIN CATCH
        SELECT 'An error occurred: ' + ERROR_MESSAGE() AS Msg;
    END CATCH
END;
```

- Report that takes the student ID and returns the grades of the student in all courses.



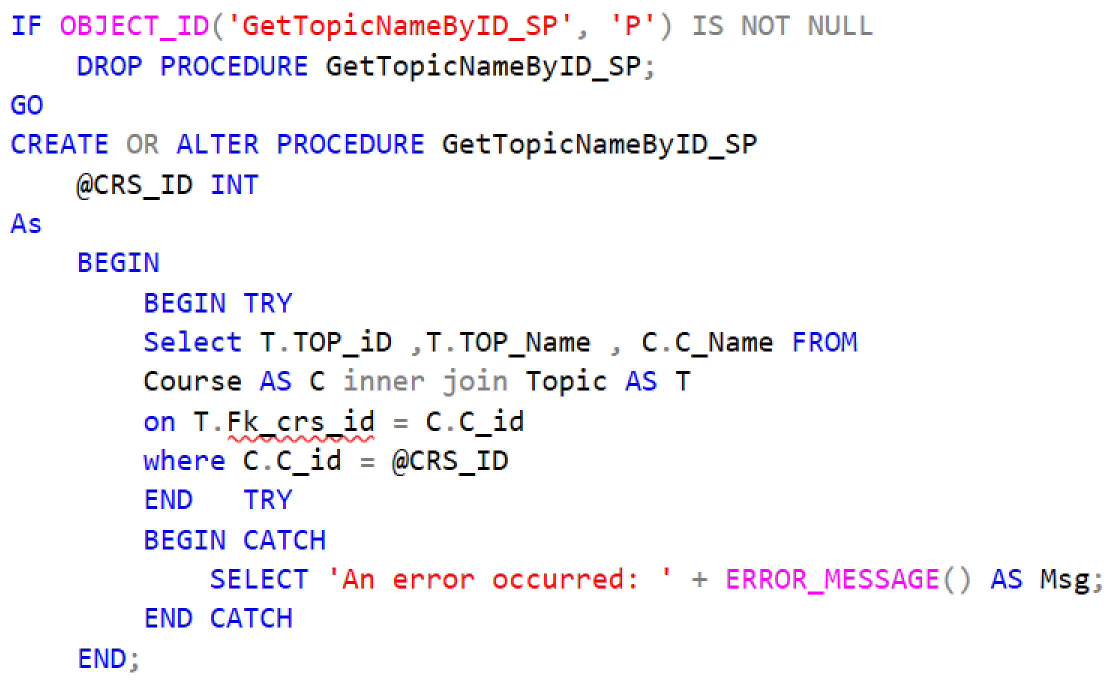
```
CREATE OR ALTER PROCEDURE GetStudentGradesAllCoures_SP
    @Std_ID INT
AS
BEGIN
    BEGIN TRY
        SELECT
            S.S_ID,
            CONCAT(S.F_Name, ' ', S.L_Name) AS StudentName,
            C.C_ID,C.C_Name,
            ISNULL(COUNT(DISTINCT E.Ex_Id), 0) AS ExamCount,
            ISNULL(SUM(CAST(SE.Grade AS FLOAT)), 0) AS TotalGrade,
            CASE
                WHEN ISNULL(SUM(CAST(SE.Grade AS FLOAT)), 0) >= (ISNULL(COUNT(DISTINCT E.Ex_Id), 0) * 10) / 2
                THEN 'Pass'
                ELSE 'Fail'
            END AS Status
        FROM Student AS S
        INNER JOIN Enroll_Crs_Std AS ES ON S.S_ID = ES.S_ID
        INNER JOIN Course AS C ON C.C_ID = ES.C_ID
        LEFT JOIN Exam AS E ON E.Fk_CID = C.C_ID
        LEFT JOIN Std_Exam AS SE ON E.Ex_Id = SE.Ex_ID AND SE.SID = S.S_ID
        WHERE S.S_ID = @Std_ID
        GROUP BY S.S_ID, S.F_Name, S.L_Name, C.C_ID, C.C_Name
        ORDER BY C.C_ID;
    END TRY
    BEGIN CATCH
        SELECT 'An error occurred: ' + ERROR_MESSAGE() AS Msg;
    END CATCH
```

➢ Report that takes the instructor's ID and returns the name of the courses that he teaches and the number of students per course.



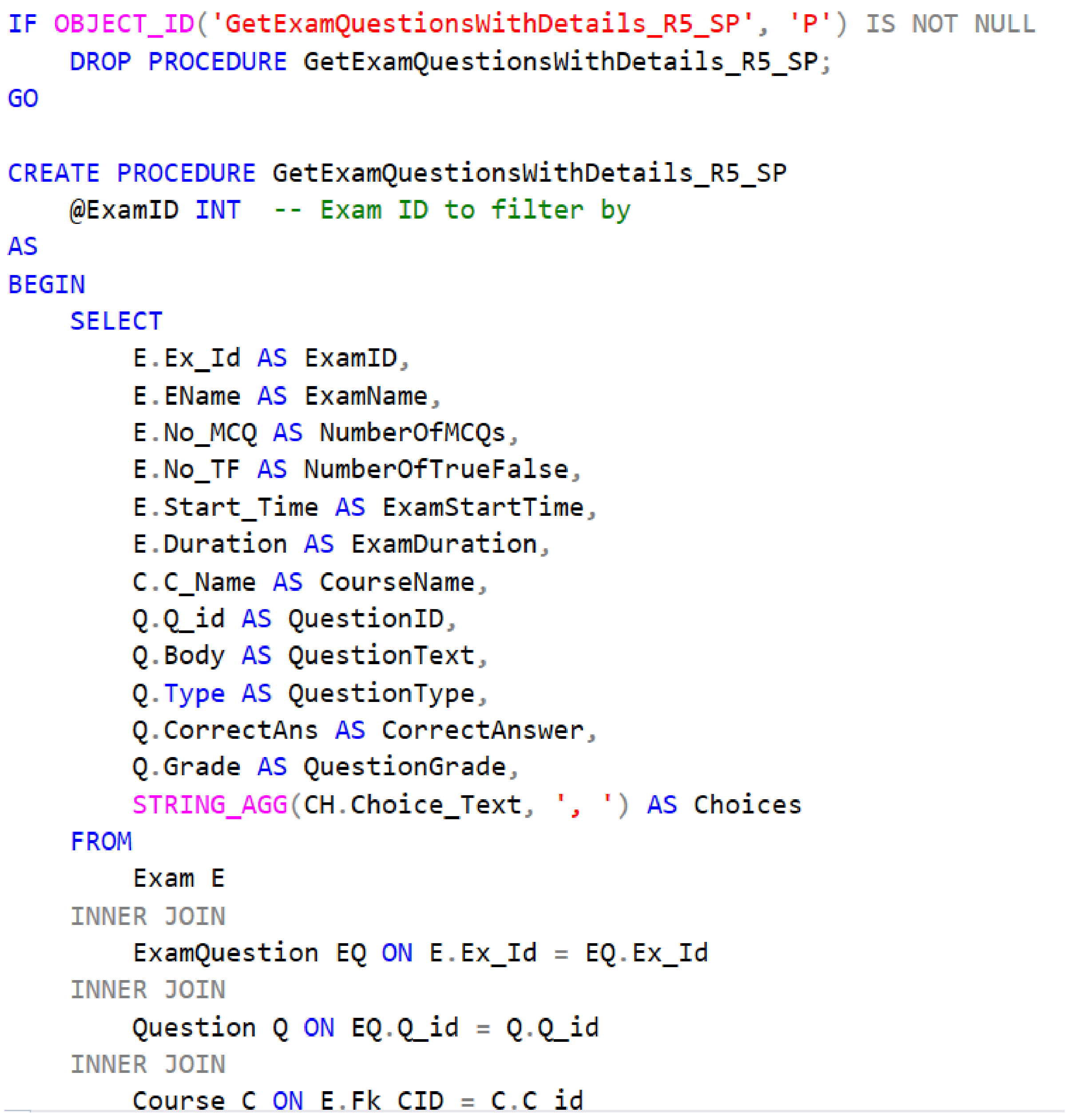
```
IF OBJECT_ID('GetCoursesAndStudentCount_SP', 'P') IS NOT NULL
    DROP PROCEDURE GetCoursesAndStudentCount_SP;
GO
CREATE PROCEDURE GetCoursesAndStudentCount_SP
    @InstructorID INT
AS
BEGIN
    BEGIN TRY
        SELECT
            CONCAT(I.Ins_Fname, ' ', I.Ins_Lname) AS Instructor_Name,
            C.C_Name AS Course_Name,COUNT(ECS.S_id) AS Number_of_Students
        FROM
            Instructor I
        JOIN
            Course C ON I.Ins_id = C.FK_Ins_id
        JOIN
            Enroll_Crs_Std ECS ON C.C_id = ECS.C_id
        WHERE
            I.Ins_id = @InstructorID
        GROUP BY
            CONCAT(I.Ins_Fname, ' ', I.Ins_Lname), C.C_Name
        ORDER BY
            COUNT(ECS.S_id) DESC;
    END TRY
    BEGIN CATCH
        PRINT 'An error occurred while fetching courses and student counts!';
        THROW;
    END CATCH;
END;
```

- Report that takes course ID and returns its topics



```
IF OBJECT_ID('GetTopicNameByID_SP', 'P') IS NOT NULL
    DROP PROCEDURE GetTopicNameByID_SP;
GO
CREATE OR ALTER PROCEDURE GetTopicNameByID_SP
    @CRS_ID INT
As
    BEGIN
        BEGIN TRY
        Select T.TOP_iD ,T.TOP_Name , C.C_Name FROM
        Course AS C inner join Topic AS T
        on T.Fk_crs_id = C.C_id
        where C.C_id = @CRS_ID
        END   TRY
        BEGIN CATCH
            SELECT 'An error occurred: ' + ERROR_MESSAGE() AS Msg;
        END CATCH
    END;
```

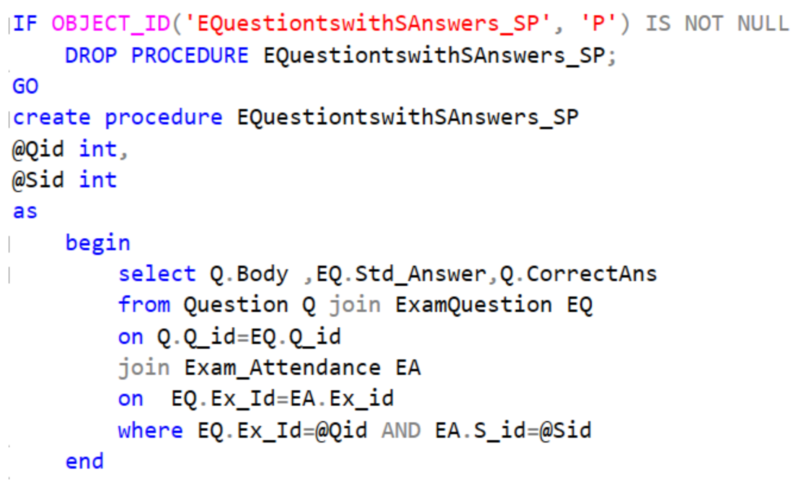
- Report that takes exam number and returns Questions in it



```sql
IF OBJECT_ID('GetExamQuestionsWithDetails_R5_SP', 'P') IS NOT NULL
    DROP PROCEDURE GetExamQuestionsWithDetails_R5_SP;
GO

CREATE PROCEDURE GetExamQuestionsWithDetails_R5_SP
    @ExamID INT  -- Exam ID to filter by
AS
BEGIN
    SELECT
        E.Ex_Id AS ExamID,
        E.EName AS ExamName,
        E.No_MCQ AS NumberOfMCQs,
        E.No_TF AS NumberOfTrueFalse,
        E.Start_Time AS ExamStartTime,
        E.Duration AS ExamDuration,
        C.C_Name AS CourseName,
        Q.Q_id AS QuestionID,
        Q.Body AS QuestionText,
        Q.Type AS QuestionType,
        Q.CorrectAns AS CorrectAnswer,
        Q.Grade AS QuestionGrade,
        STRING_AGG(CH.Choice_Text, ', ') AS Choices
    FROM
        Exam E
    INNER JOIN
        ExamQuestion EQ ON E.Ex_Id = EQ.Ex_Id
    INNER JOIN
        Question Q ON EQ.Q_id = Q.Q_id
    INNER JOIN
        Course C ON E.Fk_CID = C.C_id
```

```
   LEFT JOIN
        Choices CH ON Q.Q_id = CH.Q_id
    WHERE
        E.Ex_Id = @ExamID
    GROUP BY
        E.Ex_Id, E.EName, E.No_MCQ, E.No_TF, E.Start_Time, E.Duration,
        C.C_Name, Q.Q_id, Q.Body, Q.Type, Q.CorrectAns, Q.Grade
    ORDER BY
        Q.Q_id;
END;
GO
```

- Report that takes exam number and the student ID then returns the Questions in this exam with the student answers.

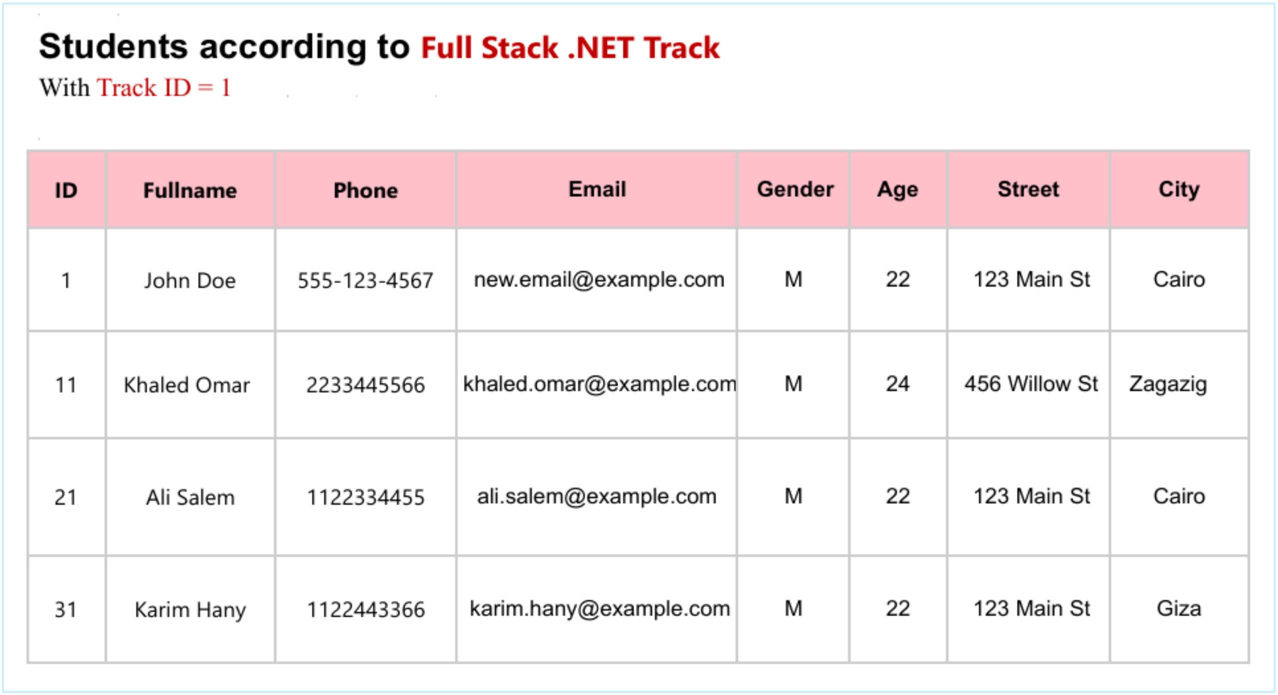


```sql
IF OBJECT_ID('EQuestiontswithSAnswers_SP', 'P') IS NOT NULL
    DROP PROCEDURE EQuestiontswithSAnswers_SP;
GO
create procedure EQuestiontswithSAnswers_SP
@Qid int,
@Sid int
as
    begin
        select Q.Body ,EQ.Std_Answer,Q.CorrectAns
        from Question Q join ExamQuestion EQ
        on Q.Q_id=EQ.Q_id
        join Exam_Attendance EA
        on  EQ.Ex_Id=EA.Ex_id
        where EQ.Ex_Id=@Qid AND EA.S_id=@Sid
    end
```

Reports

➢ Student Information Report by Department

This report is used to retrieve student details based on the Department No parameter. When a user enters a specific department number, the report returns a list of students who belong to that department, helping administrators and department heads track student records efficiently. It ensures that users can quickly find students based on their department, making data management easier.

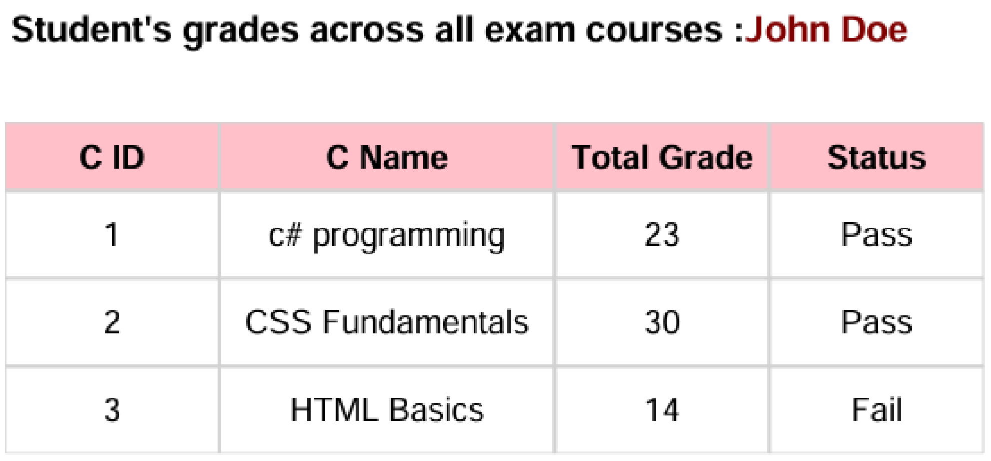


Students according to Full Stack .NET Track
With Track ID = 1

ID	Fullname	Phone	Email	Gender	Age	Street	City
1	John Doe	555-123-4567	new.email@example.com	M	22	123 Main St	Cairo
11	Khaled Omar	2233445566	khaled.omar@example.com	M	24	456 Willow St	Zagazig
21	Ali Salem	1122334455	ali.salem@example.com	M	22	123 Main St	Cairo
31	Karim Hany	1122443366	karim.hany@example.com	M	22	123 Main St	Giza

➢ Student Grades Report

This report retrieves the grades of a student in all courses based on the Student ID parameter. When a user enters a specific student ID, the report returns a list of all courses the student is enrolled in, along with the corresponding grades. This helps administrators, instructors, and students track academic performance efficiently, ensuring easy access to student records for evaluation and decision-making.

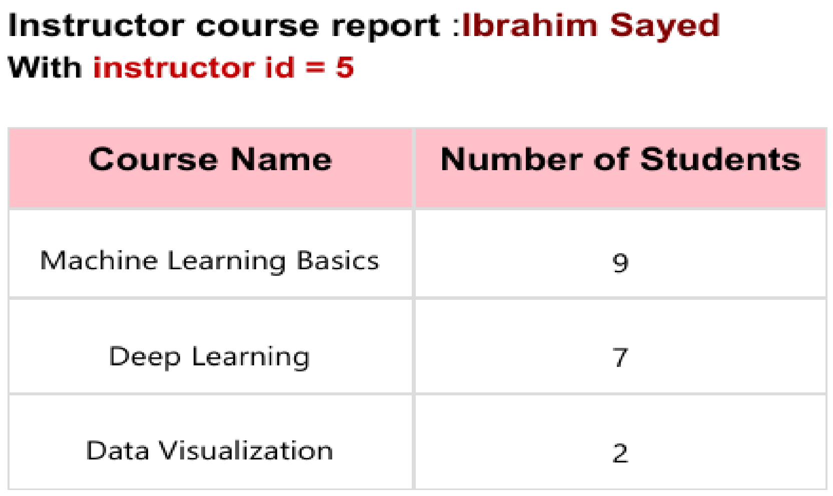


Student's grades across all exam courses :John Doe

C ID	C Name	Total Grade	Status
1	c# programming	23	Pass
2	CSS Fundamentals	30	Pass
3	HTML Basics	14	Fail

➢ Instructor Course Report

This report retrieves the list of courses taught by an instructor based on the Instructor ID parameter. When a user enters a specific instructor ID, the report returns the course names along with the number of students enrolled in each course. This helps administrators and instructors track teaching assignments and student distribution, ensuring better academic planning and resource management.

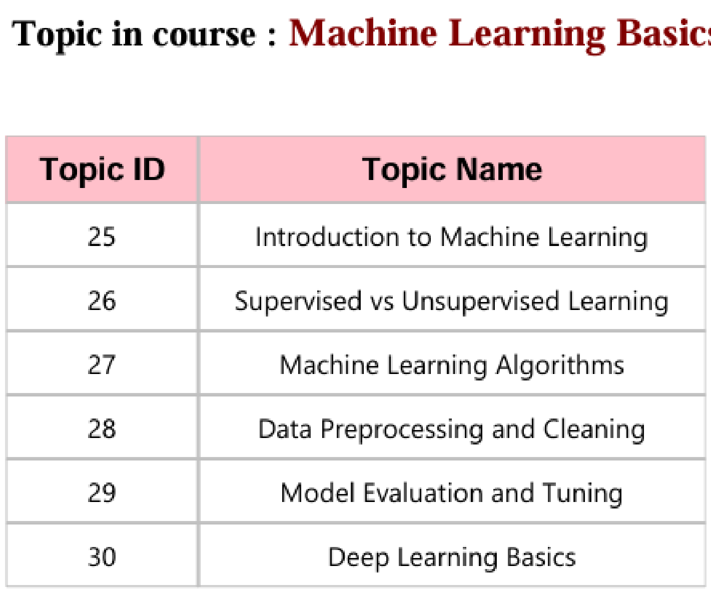


Instructor course report :Ibrahim Sayed
With instructor id = 5

Course Name	Number of Students
Machine Learning Basics	9
Deep Learning	7
Data Visualization	2

➢ Course Topics Report

This report retrieves the topics covered in a course based on the Course ID parameter. When a user enters a specific course ID, the report returns a list of topics included in that course. For example, if the course is C#, the report may show topics like Web Development, Object-Oriented Programming, and Data Structures. This helps administrators, instructors, and students understand the course structure and content more effectively.

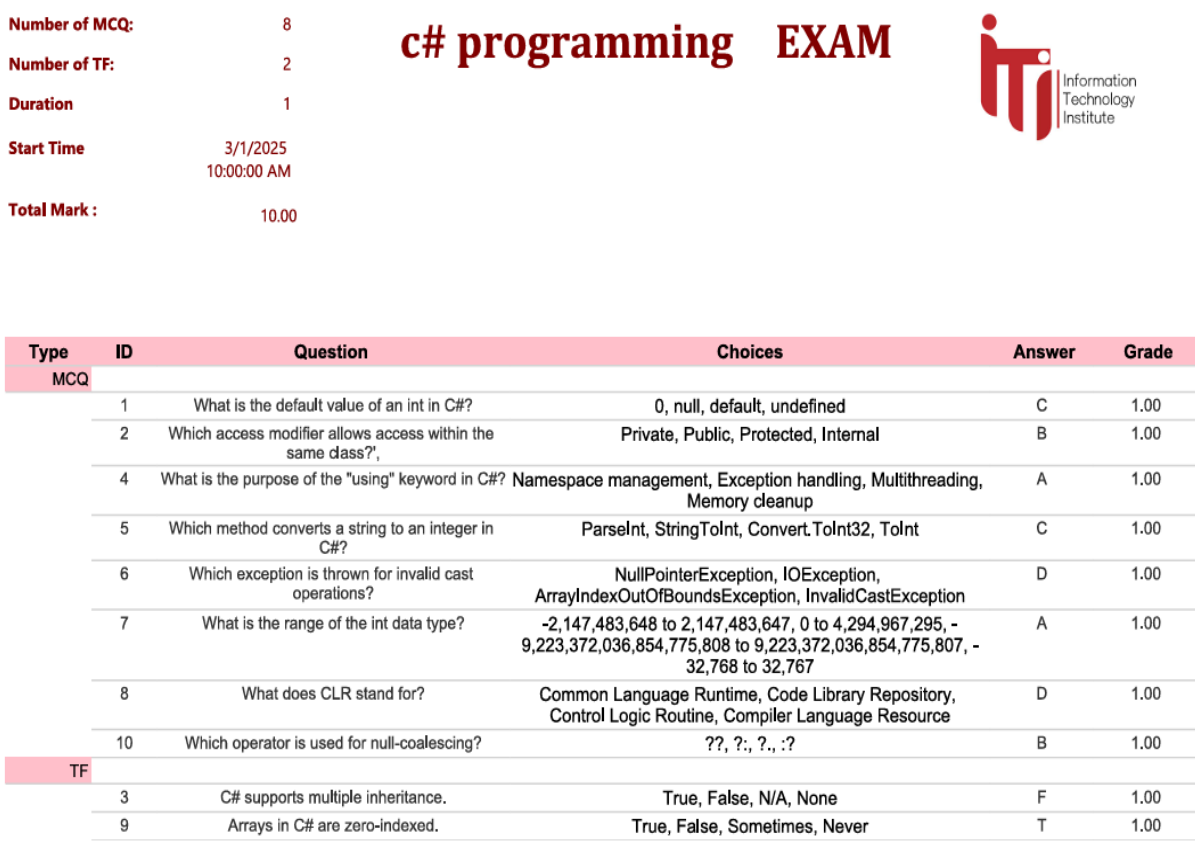


Topic in course : Machine Learning Basics

Topic ID	Topic Name
25	Introduction to Machine Learning
26	Supervised vs Unsupervised Learning
27	Machine Learning Algorithms
28	Data Preprocessing and Cleaning
29	Model Evaluation and Tuning
30	Deep Learning Basics

➢ Exam Questions Report

This report retrieves the list of questions associated with a specific exam, based on the Exam Number parameter. When a user enters a specific exam number, the report returns all the questions that are part of that exam. This report is useful for exam administrators and educators to track and manage the questions in any given exam efficiently.



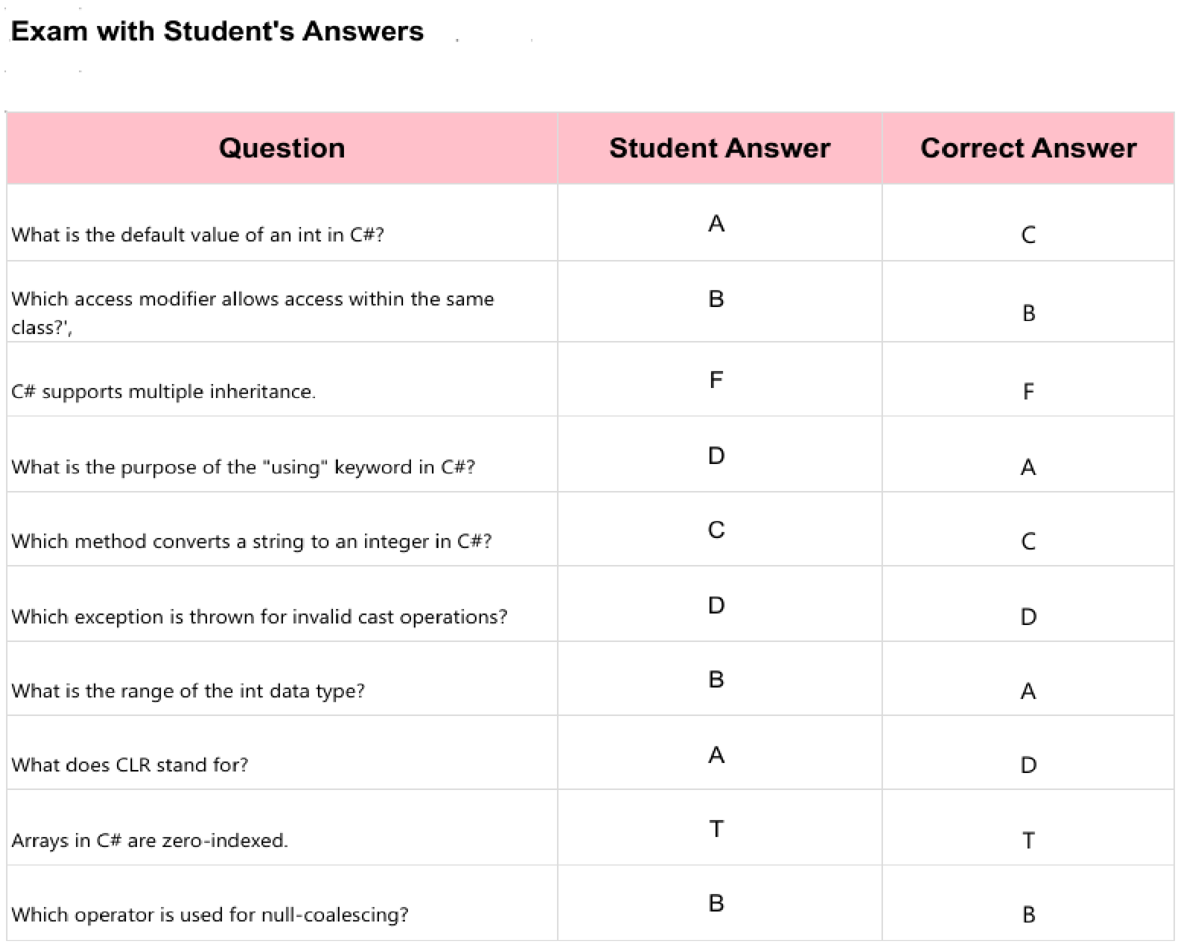
c# programming EXAM

Number of MCQ:	8
Number of TF:	2
Duration	1
Start Time	3/1/2025 10:00:00 AM
Total Mark :	10.00

Type	ID	Question	Choices	Answer	Grade
MCQ					
	1	What is the default value of an int in C#?	0, null, default, undefined	C	1.00
	2	Which access modifier allows access within the same class?',	Private, Public, Protected, Internal	B	1.00
	4	What is the purpose of the "using" keyword in C#?	Namespace management, Exception handling, Multithreading, Memory cleanup	A	1.00
	5	Which method converts a string to an integer in C#?	ParseInt, StringToInt, Convert.ToInt32, ToInt	C	1.00
	6	Which exception is thrown for invalid cast operations?	NullPointerException, IOException, ArrayIndexOutOfBoundsException, InvalidCastException	D	1.00
	7	What is the range of the int data type?	-2,147,483,648 to 2,147,483,647, 0 to 4,294,967,295, -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807, -32,768 to 32,767	A	1.00
	8	What does CLR stand for?	Common Language Runtime, Code Library Repository, Control Logic Routine, Compiler Language Resource	D	1.00
	10	Which operator is used for null-coalescing?	??, ?:, ?., :?	B	1.00
TF					
	3	C# supports multiple inheritance.	True, False, N/A, None	F	1.00
	9	Arrays in C# are zero-indexed.	True, False, Sometimes, Never	T	1.00

➢ Student Exam Answers Report

This report retrieves the list of questions and the corresponding answers provided by a specific student for a given exam, based on both the Exam Number and the Student ID parameters. When a user enters a specific exam number and student ID, the report returns all the questions in that exam along with the student's answers. This report is useful for exam administrators and educators to evaluate individual student performance and track their responses to each question in the exam.



Exam with Student's Answers

Question	Student Answer	Correct Answer
What is the default value of an int in C#?	A	C
Which access modifier allows access within the same class?',	B	B
C# supports multiple inheritance.	F	F
What is the purpose of the "using" keyword in C#?	D	A
Which method converts a string to an integer in C#?	C	C
Which exception is thrown for invalid cast operations?	D	D
What is the range of the int data type?	B	A
What does CLR stand for?	A	D
Arrays in C# are zero-indexed.	T	T
Which operator is used for null-coalescing?	B	B

Conclusion

The documentation for the Online Examination System outlines a comprehensive plan for developing an automated system to conduct online exams, supported by a robust SQL database. The system is designed to streamline the examination process for ITI staff, providing efficient exam generation, grading, and reporting functionalities. Key features include the use of stored procedures for database operations, detailed reports for student and instructor data, and integration with tools like SSRS for reporting.

The project emphasizes the importance of a well-structured database, with clear requirements for an ERD, database dictionary, and stored procedures. The system will enable ITI staff to access critical reports, such as student grades, course details, and exam results, enhancing the overall management of examinations.

Future Work

The Examination System Database, future developments can focus on the following areas:

Advanced Reporting & Analytics with Power BI

- Integration with Power BI to provide interactive dashboards and real-time analytics for student performance, instructor efficiency, and exam trends.
- Generate custom reports for tracking student progress, pass rates, and question difficulty analysis.

Automated Website & Backup System

- Develop a web-based interface for instructors and students to access exams and reports from any device.
- Implement an automatic backup system to prevent data loss and ensure disaster recovery.