Stored Procedures Documentation

This documentation provides an overview of various stored procedures available in the database.

AddCourse

```
CREATE PROCEDURE [dbo].[AddCourse]
    @crsName nvarchar(255)
AS
BEGIN
    INSERT INTO Courses VALUES (@crsName)
END
```

Usage

This stored procedure adds a new course to the Courses table.

Input Parameters

- `@crsName`: Name of the course (nvarchar, max length: 255).

Execution Command

```
EXEC [dbo].[AddCourse] @crsName = 'CourseName'
```

AddCourseDepartment

```
CREATE PROCEDURE [dbo].[addCourseDepartment]
    @courseID int,
    @deptID int
AS
BEGIN
```

```
    INSERT INTO Course_Dept VALUES (@courseID, @deptID)
END
```

Usage

This stored procedure associates a course with a department.

Input Parameters

- `@courseID`: ID of the course (int).
- `@deptID`: ID of the department (int).

Execution Command

```
EXEC [dbo].[addCourseDepartment] @courseID = 1, @deptID = 1
```

AddCourseTopic

```
CREATE PROCEDURE [dbo].[addCourseTopic]
    @courseId int,
    @topicName nvarchar(255)
AS
BEGIN
    INSERT INTO CourseTopics VALUES (@topicName, @courseId)
END
```

Usage

This stored procedure adds a topic to a specific course.

Input Parameters

- `@courseId`: ID of the course (int).
- `@topicName`: Name of the topic (nvarchar, max length: 255).

Execution Command

```
EXEC [dbo].[addCourseTopic] @courseId = 1, @topicName = 'TopicName'
```

AddInstructorCourse

```
CREATE PROCEDURE [dbo].[addInstructorCourse]
    @crsid int,
    @instructorID int
AS
BEGIN
    INSERT INTO Instructor_Courses VALUES (@crsid, @instructorID)
END
```

Usage

This stored procedure assigns an instructor to a course.

Input Parameters

- `@crsid`: ID of the course (int).
- `@instructorID`: ID of the instructor (int).

Execution Command

```
EXEC [dbo].[addInstructorCourse] @crsid = 1, @instructorID = 1
```

AddInstructorToCourse

```
CREATE PROCEDURE [dbo].[AddInstructorToCourse]
    @CourseID INT,
    @InstructorID INT
AS
```

```
BEGIN
    INSERT INTO Instructor_Courses (CourseID, InstructorID)
    VALUES (@CourseID, @InstructorID)
END
```

Usage

This stored procedure adds an instructor to a course.

Input Parameters

- `@CourseID` : ID of the course (int).
- `@InstructorID` : ID of the instructor (int).

Execution Command

```
EXEC [dbo].[AddInstructorToCourse] @CourseID = 1, @InstructorID = 1
```

AddNewBranch

```
CREATE PROCEDURE [dbo].[AddNewBranch]
    @branchName NVARCHAR(255)
AS
BEGIN
    IF LEN(@branchName) > 1
    BEGIN
        INSERT INTO Branches VALUES (@branchName)
    END
    ELSE
    BEGIN
        RAISERROR('Branch name must be at least 2 characters long.', 16, 1)
        RETURN;
    END
END
```

Usage

This stored procedure adds a new branch.

Input Parameters

- `@branchName` : Name of the branch (nvarchar, max length: 255).

Execution Command

```
EXEC [dbo].[AddNewBranch] @branchName = 'BranchName'
```

AddNewDepartment

```
CREATE PROCEDURE [dbo].[AddNewDepartment]
    @deptName nvarchar(255),
    @branchId int
AS
BEGIN
    IF LEN(@deptName) > 1
    BEGIN
        INSERT INTO Departments VALUES (@deptName, @branchId)
    END
    ELSE
    BEGIN
        RAISERROR('Department name must be at least 2 characters long.', 16, 1)
        RETURN;
    END
END
```

Usage

This stored procedure adds a new department.

Input Parameters

- `@deptName` : Name of the department (nvarchar, max length: 255).

- `@branchId`: ID of the branch to which the department belongs (int).

Execution Command

```
EXEC [dbo].[AddNewDepartment] @deptName = 'DepartmentName', @branchId = 1
```

AddQuestion

```
CREATE PROCEDURE [dbo].[AddQuestion]
    @qtext varchar(200),
    @choice1 varchar(50),
    @choice2 varchar(50),
    @choice3 varchar(50),
    @choice4 varchar(50),
    @qtyep int,
    @CorrectIndex int,
    @Grade int,
    @CourseId int
AS
BEGIN
    INSERT INTO QuestionPools VALUES (@qtext, @qtyep, @CorrectIndex, @Grade,
@CourseId);
    DECLARE @QuestId int = IDENT_CURRENT('QuestionPools');
    IF @qtyep = 1
    BEGIN
        INSERT INTO QuestionChoices VALUES (@choice1, @QuestId, 1);
        INSERT INTO QuestionChoices VALUES (@choice2, @QuestId, 2);
        INSERT INTO QuestionChoices VALUES (@choice3, @QuestId, 3);
        INSERT INTO QuestionChoices VALUES (@choice4, @QuestId, 4);
    END
    IF @qtyep = 0
    BEGIN
        INSERT INTO QuestionChoices VALUES ('true', @QuestId, 1);
        INSERT INTO QuestionChoices VALUES ('false', @QuestId, 2);
    END
END
```

Usage

This stored procedure adds a question to the question pool.

Input Parameters

- `@qtext`: Text of the question (varchar, max length: 200).
- `@choice1`: First choice for the question (varchar, max length: 50).
- `@choice2`: Second choice for the question (varchar, max length: 50).
- `@choice3`: Third choice for the question (varchar, max length: 50).
- `@choice4`: Fourth choice for the question (varchar, max length: 50).
- `@qtyep`: Quantity of choices (int).
- `@CorrectIndex`: Index of the correct choice (int).
- `@Grade`: Grade of the question (int).
- `@CourseId`: ID of the course related to the question (int).

Execution Command

```
EXEC [dbo].[AddQuestion]
    @qtext = 'Question Text',
    @choice1 = 'Choice 1',
    @choice2 = 'Choice 2',
    @choice3 = 'Choice 3',
    @choice4 = 'Choice 4',
    @qtyep = 1,
    @CorrectIndex = 1,
    @Grade = 10,
    @CourseId = 1
```

AddStudent

```
CREATE PROCEDURE [dbo].[AddStudent]
    @UserName NVARCHAR(50),
    @Email NVARCHAR(50),
    @Address NVARCHAR(100),
    @Password NVARCHAR(50),
    @Phone NVARCHAR(20),
    @Gender CHAR(1),
    @DeptID INT
AS
BEGIN
    INSERT INTO Users (UserName, Email, Address, Password, Phone, Gender)
    VALUES (@UserName, @Email, @Address, @Password, @Phone, @Gender)

    DECLARE @UserID INT
    SET @UserID = SCOPE_IDENTITY()

    INSERT INTO Students (StudentID, DeptID)
    VALUES (@UserID, @DeptID)
END
```

Usage

This stored procedure adds a new student to the database.

Input Parameters

- `@UserName` : Username of the student (nvarchar, max length: 50).
- `@Email` : Email of the student (nvarchar, max length: 50).
- `@Address` : Address of the student (nvarchar, max length: 100).
- `@Password` : Password of the student (nvarchar, max length: 50).
- `@Phone` : Phone number of the student (nvarchar, max length: 20).
- `@Gender` : Gender of the student (char, length: 1).
- `@DeptID` : ID of the department to which the student belongs (int).

Execution Command

```
EXEC [dbo].[AddStudent]
    @UserName = 'StudentName',
    @Email = 'student@example.com',
    @Address = 'Student Address',
    @Password = 'password',
    @Phone = '1234567890',
    @Gender = 'M',
    @DeptID = 1
```

AuthenticateUser

```sql
CREATE PROCEDURE [dbo].[AuthenticateUser]
    @Email NVARCHAR(255),
    @Password NVARCHAR(255),
    @UserType NVARCHAR(50) OUTPUT,
    @UserID INT OUTPUT
AS
BEGIN
    SET @UserID  = (SELECT UserID FROM [Users] WHERE Email = @Email AND Password = 
@Password);

    IF @UserID IS NULL
    BEGIN
        SET @UserType = 'Invalid';
        SET @UserID = -1;
    END
    ELSE
    BEGIN
        IF EXISTS (SELECT 1 FROM [Students] WHERE StudentID = @UserID)
        BEGIN
            SET @UserType = 'Student';
        END
        ELSE
        BEGIN
            SET @UserType = 'Instructor';
        END
    END
END
```

Usage

This stored procedure authenticates users based on their email and password.

Input Parameters

- `@Email`: Email of the user (nvarchar, max length: 255).
- `@Password`: Password of the user (nvarchar, max length: 255).

output Parameters

- `@UserType`: Type of the user (nvarchar, max length: 50).
- `@UserID`: ID of the user (int).

Execution Command

```
DECLARE @UserType NVARCHAR(50)
DECLARE @UserID INT
EXEC [dbo].[AuthenticateUser]
    @Email = 'user@example.com',
    @Password = 'password',
    @UserType = @UserType OUTPUT,
    @UserID = @UserID OUTPUT
```

---

changeCourseName

```
CREATE PROCEDURE [dbo].[changeCourseName]
    @id int,
    @name nvarchar(255)
AS
BEGIN
    UPDATE Courses SET CourseName = @name WHERE CourseID = @id
END
```

Usage

This stored procedure updates the name of a course.

Input Parameters

- `@id`: ID of the course (int).
- `@name`: New name for the course (nvarchar, max length: 255).

Execution Command

```
EXEC [dbo].[changeCourseName] @id = 1, @name = 'New Course Name'
```

DeleteAllDataFromTables

```
CREATE PROCEDURE [dbo].[DeleteAllDataFromTables]
AS
BEGIN
    DECLARE @tableName NVARCHAR(MAX)
    DECLARE tableCursor CURSOR FOR
        SELECT TABLE_NAME
        FROM INFORMATION_SCHEMA.TABLES
        WHERE TABLE_TYPE = 'BASE TABLE'
        AND TABLE_NAME NOT IN ('sysdiagrams') -- Exclude system tables if necessary

    OPEN tableCursor
    FETCH NEXT FROM tableCursor INTO @tableName

    WHILE @@FETCH_STATUS = 0
    BEGIN
        DECLARE @sql NVARCHAR(MAX)
        SET @sql = 'DELETE FROM ' + @tableName
        EXEC sp_executesql @sql

        FETCH NEXT FROM tableCursor INTO @tableName
    END

    CLOSE tableCursor
    DEALLOCATE tableCursor
END
```

Usage

This stored procedure deletes all data from the tables in the database except for system tables.

Execution Command

```
EXEC [dbo].[DeleteAllDataFromTables]
```

DeleteBranch

```
CREATE PROCEDURE [dbo].[DeleteBranch]
    @id int
AS
BEGIN
    DELETE FROM Branches WHERE BranchID = @id
END
```

Usage

This stored procedure deletes a branch from the database.

Input Parameters

- `@id`: ID of the branch to be deleted (int).

Execution Command

```
EXEC [dbo].[DeleteBranch] @id = 1
```

---

deleteCourseDepartment

```
CREATE PROCEDURE [dbo].[deleteCourseDepartment]
    @courseID int,
    @deptID int
AS
BEGIN
    DELETE FROM Course_Dept WHERE CourseID = @courseID AND DeptId = @deptID
END
```

Usage

This stored procedure deletes a course from a department.

Input Parameters

- `@courseID`: ID of the course (int).
- `@deptID`: ID of the department (int).

Execution Command

```
EXEC [dbo].[deleteCourseDepartment] @courseID = 1, @deptID = 1
```

---

deleteCourseInstructorByInstructorIDAndCourseID

```
CREATE PROCEDURE [dbo].[deleteCourseInstructorByInstructorIDAndCourseID]
    @crsid int,
    @instructorID int
AS
BEGIN
    DELETE FROM Instructor_Courses WHERE CourseID = @crsid AND InstructorID =
@instructorID
END
```

Usage

This stored procedure deletes an instructor from a course.

Input Parameters

- `@crsid`: ID of the course (int).
- `@instructorID`: ID of the instructor (int).

Execution Command

```
EXEC [dbo].[deleteCourseInstructorByInstructorIDAndCourseID] @crsid = 1,
@instructorID = 1
```

---

deleteCourseTopic

```
CREATE PROCEDURE [dbo].[deleteCourseTopic]
    @courseId int,
    @topicName nvarchar(255)
AS
BEGIN
    DELETE FROM CourseTopics WHERE CourseID = @courseId AND Topic = @topicName
END
```

Usage

This stored procedure deletes a topic from a course.

Input Parameters

- `@courseId`: ID of the course (int).
- `@topicName`: Name of the topic (nvarchar, max length: 255).

Execution Command

```
EXEC [dbo].[deleteCourseTopic] @courseId = 1, @topicName = 'TopicName'
```

DeleteDepartmentById

```
CREATE PROCEDURE [dbo].[DeleteDepartmentById]
    @id int
AS
BEGIN
    DELETE FROM Departments WHERE DeptID = @id
END
```

Usage

This stored procedure deletes a department by its ID.

Input Parameters

- `@id`: ID of the department (int).

Execution Command

```
EXEC [dbo].[DeleteDepartmentById] @id = 1
```

---

deleteExam

```
CREATE PROCEDURE [dbo].[deleteExam]
    @eid int
AS
BEGIN
    DELETE FROM Exams WHERE ExamID = @eid
END
```

Usage

This stored procedure deletes an exam by its ID.

Input Parameters

- `@eid`: ID of the exam (int).

Execution Command

```
EXEC [dbo].[deleteExam] @eid = 1
```

DeleteInstructor

```
CREATE PROCEDURE [dbo].[DeleteInstructor]
    @UserID INT
AS
BEGIN
```

```sql
    DELETE FROM Instructors WHERE InstructorID = @UserID
    DELETE FROM Users WHERE UserID = @UserID
END
```

Usage

This stored procedure deletes an instructor from the database along with their user account.

Input Parameters

- `@UserID`: ID of the instructor (int).

Execution Command

```sql
EXEC [dbo].[DeleteInstructor] @UserID = 1
```

EditInstructor

```sql
CREATE PROCEDURE [dbo].[EditInstructor]
    @UserID INT,
    @UserName NVARCHAR(50),
    @Email NVARCHAR(50),
    @Address NVARCHAR(100),
    @Password NVARCHAR(50),
    @Phone NVARCHAR(20),
    @Gender CHAR(1),
    @BranchID INT
AS
BEGIN
    IF EXISTS (SELECT * FROM Users WHERE UserID = @UserID)
    BEGIN
        EXEC UpdateInstructor @UserID, @UserName, @Email, @Address, @Password,
@Phone, @Gender, @BranchID
    END
    ELSE
    BEGIN
        EXEC AddInstructor @UserName, @Email, @Address, @Password, @Phone, @Gender,
@BranchID
    END
END
```

Usage

This stored procedure edits an existing instructor's details or adds a new instructor if the user does not exist.

Input Parameters

- `@UserID`: ID of the instructor (int).
- `@UserName`: Username of the instructor (nvarchar, max length: 50).
- `@Email`: Email of the instructor (nvarchar, max length: 50).
- `@Address`: Address of the instructor (nvarchar, max length: 100).
- `@Password`: Password of the instructor (nvarchar, max length: 50).
- `@Phone`: Phone number of the instructor (nvarchar, max length: 20).
- `@Gender`: Gender of the instructor (char, length: 1).
- `@BranchID`: ID of the branch where the instructor belongs (int).

Execution Command

```
EXEC [dbo].[EditInstructor]
    @UserID = 1,
    @UserName = 'InstructorName',
    @Email = 'instructor@example.com',
    @Address = 'Instructor Address',
    @Password = 'password',
    @Phone = '1234567890',
    @Gender = 'M',
    @BranchID = 1
```

EditStudent

```
CREATE PROCEDURE [dbo].[EditStudent]
    @UserID INT,
    @UserName NVARCHAR(50),
    @Email NVARCHAR(50),
    @Address NVARCHAR(100),
    @Password NVARCHAR(50),
    @Phone NVARCHAR(20),
    @Gender CHAR(1),
    @DeptID INT
AS
BEGIN
    IF EXISTS (SELECT * FROM Users WHERE UserID = @UserID)
    BEGIN
        EXEC UpdateStudent @UserID, @UserName, @Email, @Address, @Password, @Phone,
@Gender, @DeptID
    END
    ELSE
    BEGIN
        EXEC AddStudent @UserName, @Email, @Address, @Password, @Phone, @Gender,
@DeptID
    END
END
```

Usage

This stored procedure edits an existing student's details or adds a new student if the user does not exist.

Input Parameters

- `@UserID`: ID of the student (int).
- `@UserName`: Username of the student (nvarchar, max length: 50).
- `@Email`: Email of the student (nvarchar, max length: 50).
- `@Address`: Address of the student (nvarchar, max length: 100).
- `@Password`: Password of the student (nvarchar, max length: 50).
- `@Phone`: Phone number of the student (nvarchar, max length: 20).
- `@Gender`: Gender of the student (char, length: 1).
- `@DeptID`: ID of the department where the student belongs (int).

Execution Command

```
EXEC [dbo].[EditStudent]
    @UserID = 1,
    @UserName = 'StudentName',
    @Email = 'student@example.com',
    @Address = 'Student Address',
    @Password = 'password',
    @Phone = '1234567890',
    @Gender = 'M',
    @DeptID = 1
```

ExamReport

```
CREATE PROCEDURE [dbo].[ExamReport]
    @examid int
AS
BEGIN
    SELECT qp.Title, qc.Choice
    FROM Exams e
    JOIN ExamQuestions eq ON eq.ExamID = e.ExamID
    JOIN QuestionPools qp ON qp.QuestionID = eq.QuestionID
    JOIN QuestionChoices qc ON qc.QuestionID = qp.QuestionID
    WHERE e.ExamID = @examid;
END
```

Usage

This stored procedure generates a report for an exam showing the questions and choices.

Input Parameters

- `@examid`: ID of the exam (int).

Execution Command

```
EXEC [dbo].[ExamReport] @examid = 1
```

GenerateExam

```sql
CREATE PROCEDURE [dbo].[GenerateExam]
    @courseId int,
    @deptId int,
    @date datetime,
    @mcqCount int,
    @TFcount int,
    @duration int
AS
BEGIN
    DECLARE @ExamId int;
    SELECT @ExamId = IDENT_CURRENT('Exams') + 1;
    INSERT INTO Exams VALUES (@date, @duration, @courseId, @deptId);
    EXEC GetRandomQuestions @mcqCount, @TFcount, @courseId, @ExamId;
END
```

Usage

This stored procedure generates an exam by randomly selecting questions from the question pool.

Input Parameters

- `@courseId` : ID of the course (int).
- `@deptId` : ID of the department (int).
- `@date` : Date of the exam (datetime).
- `@mcqCount` : Number of multiple-choice questions to include in the exam (int).
- `@TFcount` : Number of true/false questions to include in the exam (int).
- `@duration` : Duration of the exam in minutes (int).

Execution Command

```sql
EXEC [dbo].[GenerateExam]
    @courseId = 1,
    @deptId = 1,
    @date = '2024-03-26 09:00:00',
    @mcqCount = 10,
    @TFcount = 5,
    @duration = 90
```

GetAllBranches

```
CREATE PROCEDURE [dbo].[GetAllBranches]
AS
BEGIN
    SELECT * FROM Branches
END
```

Usage

This stored procedure retrieves all branches from the database.

Execution Command

```
EXEC [dbo].[GetAllBranches]
```

---

getAllCourses

```
CREATE PROCEDURE [dbo].[getAllCourses]
AS
BEGIN
    SELECT * FROM Courses
END
```

Usage

This stored procedure retrieves all courses from the database.

Execution Command

```sql
EXEC [dbo].[getAllCourses]
```

GetAllCoursesInBranchByInstructorID

```sql
CREATE PROCEDURE [dbo].[GetAllCoursesInBranchByInstructorID]
    @InstructorID INT
AS
BEGIN
    DECLARE @BranchID INT

    EXEC GetBranchIDByInstructorID @InstructorID, @BranchID OUTPUT

    SELECT DISTINCT Branches.BranchID, Courses.CourseID, Courses.CourseName
    FROM Courses
    INNER JOIN Course_Dept ON Courses.CourseID = Course_Dept.CourseID
    INNER JOIN Departments ON Course_Dept.DeptId = Departments.DeptID
    INNER JOIN Branches ON Departments.BranchID = Branches.BranchID
    WHERE Branches.BranchID = @BranchID
END
```

Usage

This stored procedure retrieves all courses in a branch associated with a given instructor ID.

Input Parameters

- `@InstructorID`: ID of the instructor (int).

Execution Command

```sql
EXEC [dbo].[GetAllCoursesInBranchByInstructorID] @InstructorID = 1
```

GetAllDepartments

```sql
CREATE PROCEDURE [dbo].[GetAllDepartments]
AS
BEGIN
    SELECT * FROM Departments
END
```

Usage

This stored procedure retrieves all departments from the database.

Execution Command

```sql
EXEC [dbo].[GetAllDepartments]
```

GetAllDeptCourses

```sql
CREATE PROCEDURE [dbo].[GetAllDeptCourses]
    @deptid int
AS
BEGIN
    SELECT c.CourseID, c.CourseName
    FROM Courses c
    JOIN Course_Dept dc ON dc.CourseID = c.CourseID
    JOIN Departments d ON d.DeptId = dc.DeptId
    WHERE d.DeptID = @deptid
END
```

Usage

This stored procedure retrieves all courses in a department based on the department ID.

Input Parameters

- `@deptid`: ID of the department (int).

Execution Command

```
EXEC [dbo].[GetAllDeptCourses] @deptid = 1
```

---

getAllExamQuestions

```
Create PROCEDURE [dbo].[getAllExamQuestions]
AS
BEGIN
    SELECT * FROM ExamQuestions;
END
```

Usage

This stored procedure retrieves all exam questions from the database.

Execution Command

```
EXEC [dbo].[getAllExamQuestions]
```

---

getAllExams

```
CREATE PROCEDURE [dbo].[getAllExams]
AS
BEGIN
    SELECT * FROM Exams
END
```

Usage

This stored procedure retrieves all exams from the database.

Execution Command

```
EXEC [dbo].[getAllExams]
```

---

getallinstructors

```
CREATE PROCEDURE [dbo].[getallinstructors]
AS
BEGIN
    SELECT u.*
    FROM Instructors i
    INNER JOIN Users u ON i.InstructorID = u.UserID
END
```

Usage

This stored procedure retrieves all instructors from the database along with their user details.

Execution Command

```
EXEC [dbo].[getallinstructors]
```

---

getAllQuestionChoices

```
CREATE PROCEDURE [dbo].[getAllQuestionChoices]
AS
BEGIN
    SELECT * FROM QuestionChoices
END
```

Usage

This stored procedure retrieves all question choices from the database.

Execution Command

```
EXEC [dbo].[getAllQuestionChoices]
```

---

getAllQuestionPools

```
CREATE PROCEDURE [dbo].[getAllQuestionPools]
AS
BEGIN
    SELECT * FROM QuestionPools
END
```

Usage

This stored procedure retrieves all question pools from the database.

Execution Command

```
EXEC [dbo].[getAllQuestionPools]
```

GetAllQuestions

```
CREATE PROCEDURE [dbo].[GetAllQuestions]
    @crsId int
AS
BEGIN
    DECLARE @Isselected bit = 0;
    SELECT qp.QuestionID, isSelected = @Isselected, qp.Title,
qp.CorrectAnswerIndex, qp.Grade, qc.Choice
    FROM QuestionPools qp
    JOIN QuestionChoices qc ON qc.QuestionID = qp.QuestionID AND qc.ChoiceIndex =
qp.CorrectAnswerIndex
    WHERE CourseID = @crsId
END
```

Usage

This stored procedure retrieves all questions associated with a particular course.

Input Parameters

- `@crsId`: ID of the course (int).

Execution Command

```
EXEC [dbo].[GetAllQuestions] @crsId = 1
```

---

getAllStudentExamQuestions

```
CREATE PROCEDURE [dbo].[getAllStudentExamQuestions]
AS
BEGIN
    SELECT * FROM StudentExamQuestions;
END
```

Usage

This stored procedure retrieves all student exam questions from the database.

Execution Command

```
EXEC [dbo].[getAllStudentExamQuestions]
```

---

getAllStudents

```
CREATE PROCEDURE [dbo].[getAllStudents]
AS
BEGIN
    SELECT * FROM Students
END
```

Usage

This stored procedure retrieves all students from the database.

Execution Command

```
EXEC [dbo].[getAllStudents]
```

---

getallusers

```
CREATE PROCEDURE [dbo].[getallusers]
AS
BEGIN
    SELECT * FROM Users
END
```

Usage

This stored procedure retrieves all users from the database.

Execution Command

```
EXEC [dbo].[getallusers]
```

---

getBranchByName

```sql
CREATE PROCEDURE [dbo].[getBranchByName]
    @name nvarchar(255)
AS
BEGIN
    SELECT * FROM Branches WHERE LOWER(BranchName) = LOWER(@name)
END
```

Usage

This stored procedure retrieves branches by name.

Input Parameters

- `@name`: Name of the branch (nvarchar).

Execution Command

```sql
EXEC [dbo].[getBranchByName] @name = 'branch_name'
```

GetBrancheByID

```sql
CREATE PROCEDURE [dbo].[GetBrancheByID]
    @branchId int
AS
BEGIN
    SELECT * FROM Branches WHERE BranchID = @branchId
END
```

Usage

This stored procedure retrieves a branch by its ID.

Input Parameters

- `@branchId`: ID of the branch (int).

Execution Command

```
EXEC [dbo].[GetBrancheByID] @branchId = 1
```

---

getBranchesDepartments

```
CREATE PROCEDURE [dbo].[getBranchesDepartments]
    @id int
AS
BEGIN
    SELECT d.*
    FROM Branches b INNER JOIN Departments d
    ON d.BranchID = @id
END
```

Usage

This stored procedure retrieves departments of a branch by branch ID.

Input Parameters

- `@id`: ID of the branch (int).

Execution Command

```
EXEC [dbo].[getBranchesDepartments] @id = 1
```

GetBranchIDByInstructorID

```
CREATE PROCEDURE [dbo].[GetBranchIDByInstructorID]
    @InstructorID INT,
    @BranchID INT OUTPUT
AS
BEGIN
    SELECT @BranchID = BranchID
    FROM Instructors
    WHERE InstructorID = @InstructorID
END
```

Usage

This stored procedure retrieves the branch ID associated with a given instructor ID.

Input Parameters

- `@InstructorID`: ID of the instructor (int).

output Parameters

- `@BranchID`: Output parameter to store the branch ID associated with the instructor.

Execution Command

```
DECLARE @BranchID INT
EXEC [dbo].[GetBranchIDByInstructorID] @InstructorID = 1, @BranchID OUTPUT
PRINT @BranchID
```

---

getBranchName

```
CREATE PROCEDURE [dbo].[getBranchName]
    @id int
AS
BEGIN
    SELECT BranchName FROM Branches WHERE BranchID=@id
END
```

Usage

This stored procedure retrieves the name of a branch by its ID.

Input Parameters

- `@id`: ID of the branch (int).

Execution Command

```
EXEC [dbo].[getBranchName] @id = 1
```

---

getCourseByName

```
CREATE PROCEDURE [dbo].[getCourseByName]
    @name nvarchar(255)
AS
BEGIN
    SELECT * FROM Courses WHERE CourseName=@name
END
```

Usage

This stored procedure retrieves a course by its name.

Input Parameters

- `@name`: Name of the course (nvarchar).

Execution Command

```
EXEC [dbo].[getCourseByName] @name = 'course_name'
```

---

getCourseDepartment

```
CREATE PROCEDURE [dbo].[getCourseDepartment]
    @courseID int,
    @deptID int
AS
BEGIN
    SELECT * FROM Course_Dept WHERE DeptId = @deptID AND CourseID=@courseID
END
```

Usage

This stored procedure retrieves a course department by course ID and department ID.

Input Parameters

- `@courseID`: ID of the course (int).
- `@deptID`: ID of the department (int).

Execution Command

```
EXEC [dbo].[getCourseDepartment] @courseID = 1, @deptID = 1
```

---

getCourseDepartments

```
CREATE PROCEDURE [dbo].[getCourseDepartments]
    @id int
AS
BEGIN
    SELECT * FROM Course_Dept WHERE CourseID=@id;
END
```

Usage

This stored procedure retrieves departments associated with a specific course.

Input Parameters

- `@id`: ID of the course (int).

Execution Command

```
EXEC [dbo].[getCourseDepartments] @id = 1
```

---

getCourseDepartmentsAllInfo

```
CREATE PROCEDURE [dbo].[getCourseDepartmentsAllInfo]
    @courseId int
AS
BEGIN
    SELECT d.*
    FROM Course_Dept cp
    INNER JOIN Departments d ON cp.DeptId = d.DeptID
    WHERE CourseID=@courseId;
END
```

Usage

This stored procedure retrieves all information about departments associated with a specific course.

Input Parameters

- `@courseId`: ID of the course (int).

Execution Command

```
EXEC [dbo].[getCourseDepartmentsAllInfo] @courseId = 1
```

GetCourseExams

```
CREATE PROCEDURE [dbo].[GetCourseExams]
    @DeptID INT
AS
BEGIN
    SELECT Exams.ExamID, Courses.CourseName, Exams.ExamDate, Exams.Duration
    FROM Courses
    INNER JOIN Exams ON Courses.CourseID = Exams.CourseID
    WHERE Exams.DeptID = @DeptID;
END
```

Usage

This stored procedure retrieves exams associated with a specific department.

Input Parameters

- `@DeptID`: ID of the department (int).

Execution Command

```
EXEC [dbo].[GetCourseExams] @DeptID = 1
```

---

getCourseIdByExamID

```
CREATE PROCEDURE [dbo].[getCourseIdByExamID]
    @ExamID int
AS
BEGIN
    SELECT CourseID FROM Exams  WHERE ExamID=@ExamID
END
```

Usage

This stored procedure retrieves the course ID associated with a given exam ID.

Input Parameters

- `@ExamID`: ID of the exam (int).

Execution Command

```
EXEC [dbo].[getCourseIdByExamID] @ExamID = 1
```

GetCourseInfo

```
CREATE PROCEDURE [dbo].[GetCourseInfo]
    @CourseID INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT CourseName, ExamDate
    FROM Courses
    INNER JOIN Exams ON Courses.CourseID = Exams.CourseID
    WHERE Courses.CourseID = @CourseID;
END
```

Usage

This stored procedure retrieves the course name and exam date associated with a given course ID.

Input Parameters

- `@CourseID`: ID of the course (int).

Execution Command

```
EXEC [dbo].[GetCourseInfo] @CourseID = 1
```

---

getCourseInstructors

```
CREATE PROCEDURE [dbo].[getCourseInstructors]
    @id int
AS
BEGIN
    SELECT * FROM Instructor_Courses WHERE CourseID=@id;
END
```

Usage

This stored procedure retrieves instructors associated with a specific course.

Input Parameters

- `@id`: ID of the course (int).

Execution Command

```
EXEC [dbo].[getCourseInstructors] @id = 1
```

---

getCourseInstructorsByCourseID

```
CREATE PROCEDURE [dbo].[getCourseInstructorsByCourseID]
    @id int
AS
BEGIN
    SELECT * FROM Instructor_Courses WHERE CourseID = @id
END
```

Usage

This stored procedure retrieves instructors associated with a specific course by course ID.

Input Parameters

- `@id`: ID of the course (int).

Execution Command

```
EXEC [dbo].[getCourseInstructorsByCourseID] @id = 1
```

---

getCourseInstructorsInfo

```
CREATE PROCEDURE [dbo].[getCourseInstructorsInfo]
    @courseId nvarchar(255)
AS
BEGIN
    SELECT i.*
    FROM Instructor_Courses ic
    INNER JOIN Users i ON ic.InstructorID = i.UserID
    WHERE ic.CourseID = @courseId
END
```

Usage

This stored procedure retrieves instructor information associated with a specific course.

Input Parameters

- `@courseId`: ID of the course (nvarchar).

Execution Command

```
EXEC [dbo].[getCourseInstructorsInfo] @courseId = 'course_id'
```

GetCourseList

```
CREATE PROCEDURE [dbo].[GetCourseList]
    @InsId INT
AS
BEGIN
    SELECT c.CourseName, c.CourseID
    FROM Courses c
    INNER JOIN Instructor_Courses IC ON c.CourseID = ic.CourseID
    WHERE IC.InstructorID = @InsId
END
```

Usage

This stored procedure retrieves a list of courses associated with a specific instructor.

Input Parameters

- `@InsId`: ID of the instructor (int).

Execution Command

```
EXEC [dbo].[GetCourseList] @InsId = 1
```

GetCoursesByInstructorID

```
CREATE PROCEDURE [dbo].[GetCoursesByInstructorID]
    @InstructorID INT
AS
BEGIN
    SELECT Courses.CourseID, Courses.CourseName
    FROM Courses
    INNER JOIN Instructor_Courses ON Courses.CourseID = Instructor_Courses.CourseID
    WHERE Instructor_Courses.InstructorID = @InstructorID
END
```

Usage

This stored procedure retrieves courses associated with a specific instructor by instructor ID.

Input Parameters

- `@InstructorID`: ID of the instructor (int).

Execution Command

```
EXEC [dbo].[GetCoursesByInstructorID] @InstructorID = 1
```

GetCoursesInBranch

```
CREATE PROCEDURE [dbo].[GetCoursesInBranch]
    @BranchID INT
AS
BEGIN
    SELECT DISTINCT Branches.BranchID, Courses.CourseID, Courses.CourseName
    FROM Courses
    INNER JOIN Course_Dept ON Courses.CourseID = Course_Dept.CourseID
    INNER JOIN Departments ON Course_Dept.DeptId = Departments.DeptId
    INNER JOIN Branches ON Departments.BranchID = Branches.BranchID
    WHERE Branches.BranchID = @BranchID
END
```

Usage

This stored procedure retrieves courses offered in a specific branch.

Input Parameters

- `@BranchID`: ID of the branch (int).

Execution Command

```
EXEC [dbo].[GetCoursesInBranch] @BranchID = 1
```

GetCoursesIns

```
CREATE PROCEDURE [dbo].[GetCoursesIns]
    @insId int
AS
BEGIN
    SELECT c.CourseID , c.CourseName
    FROM Courses c
    JOIN Instructor_Courses ic ON ic.CourseID = c.CourseID
    WHERE ic.InstructorID = @insId
END
```

Usage

This stored procedure retrieves courses associated with a specific instructor.

Input Parameters

- `@insId`: ID of the instructor (int).

Execution Command

```
EXEC [dbo].[GetCoursesIns] @insId = 1
```

---

getCourseTopics

```
CREATE PROCEDURE [dbo].[getCourseTopics]
    @id int
AS
```

```
BEGIN
    SELECT * FROM CourseTopics WHERE CourseID=@id;
END
```

Usage

This stored procedure retrieves topics associated with a specific course.

Input Parameters

- `@id` : ID of the course (int).

Execution Command

```
EXEC [dbo].[getCourseTopics] @id = 1
```

GetDepartmentByNameAndBranchID

```
CREATE PROCEDURE [dbo].[GetDepartmentByNameAndBranchID]
    @name nvarchar(255),
    @id int
AS
BEGIN
    SELECT * FROM Departments WHERE LOWER(DeptName) = Lower(@name) AND BranchID =
@id
END
```

Usage

This stored procedure retrieves a department by its name and branch ID.

Input Parameters

- `@name` : Name of the department (nvarchar).
- `@id` : ID of the branch (int).

Execution Command

```
EXEC [dbo].[GetDepartmentByNameAndBranchID] @name = 'dept_name', @id = 1
```

GetDeptsforIns

```
CREATE PROCEDURE [dbo].[GetDeptsforIns]
    @insId int
AS
BEGIN
    SELECT d.DeptName , d.DeptID, d.BranchID
    FROM Departments d
    JOIN Branches b ON d.BranchID = b.BranchID
    JOIN Instructors I ON I.BranchID = b.BranchID
    WHERE i.InstructorID = @insId
END
```

Usage

This stored procedure retrieves departments associated with a specific instructor.

Input Parameters

- `@insId`: ID of the instructor (int).

Execution Command

```
EXEC [dbo].[GetDeptsforIns] @insId = 1
```

---

getExam

```
CREATE PROCEDURE [dbo].[getExam]
    @deptid int,
```

```sql
    @crsid int
AS
BEGIN
    SELECT * FROM Exams WHERE DeptID = @deptid AND CourseID = @crsid
END
```

Usage

This stored procedure retrieves exams based on department ID and course ID.

Input Parameters

- `@deptid` : Department ID (int).
- `@crsid` : Course ID (int).

Execution Command

```sql
EXEC [dbo].[getExam] @deptid = 1, @crsid = 1
```

---

getExambyId

```sql
CREATE PROCEDURE [dbo].[getExambyId]
    @id int
AS
BEGIN
    SELECT * FROM Exams WHERE ExamID = @id
END
```

Usage

This stored procedure retrieves an exam based on its ID.

Input Parameters

- `@id` : Exam ID (int).

Execution Command

```
EXEC [dbo].[getExambyId] @id = 1
```

GetExamForStdReport

```
CREATE PROCEDURE [dbo].[GetExamForStdReport]
    @ExamID INT,
    @StudentID INT
AS
BEGIN
    SELECT
        qp.Title AS Question,
        qc.Choice AS [Correct Answer],
        qc_student.Choice AS StudentAnswer,
        CASE
            WHEN seq.SelectedAnswerIndex = qp.CorrectAnswerIndex THEN 'Correct'
            ELSE 'Incorrect'
        END AS AnswerStatus
    FROM
        QuestionPools qp
    INNER JOIN
        ExamQuestions eq ON eq.QuestionID = qp.QuestionID
    INNER JOIN
        QuestionChoices qc ON eq.QuestionID = qc.QuestionID
                        AND qc.ChoiceIndex = qp.CorrectAnswerIndex
    INNER JOIN
        StudentExamQuestions seq ON eq.ExamID = seq.ExamID
                               AND eq.QuestionID = seq.QuestionID
                               AND seq.StudentID = @StudentID
    LEFT JOIN
        QuestionChoices qc_student ON seq.QuestionID = qc_student.QuestionID
                                  AND seq.SelectedAnswerIndex =
qc_student.ChoiceIndex
    WHERE
        eq.ExamID = @ExamID;
END;
```

Usage

This stored procedure retrieves exam details for a student's report.

Input Parameters

- `@ExamID`: ID of the exam (int).
- `@StudentID`: ID of the student (int).

Execution Command

```
EXEC [dbo].[GetExamForStdReport] @ExamID = 1, @StudentID = 1
```

GetExamGradesByStudentID

```
CREATE PROCEDURE [dbo].[GetExamGradesByStudentID]
    @StudentID INT
AS
BEGIN
    SELECT Exams.ExamID, Courses.CourseName, Exams.ExamDate, StudentGrades.Grade
    FROM Courses
    INNER JOIN Exams ON Courses.CourseID = Exams.CourseID
    INNER JOIN StudentGrades ON Courses.CourseID = StudentGrades.CourseID
    WHERE StudentGrades.StudentID = @StudentID
END;
```

Usage

This stored procedure retrieves exam grades for a student based on their ID.

Input Parameters

- `@StudentID`: ID of the student (int).

Execution Command

```
EXEC [dbo].[GetExamGradesByStudentID] @StudentID = 1
```

GetExamInfo

```
CREATE PROCEDURE [dbo].[GetExamInfo]
    @ExamId INT,
    @StdId INT
AS
BEGIN
    DECLARE @CourseName NVARCHAR(100)
    DECLARE @Grade FLOAT

    -- Retrieve Course Name
    SELECT @CourseName = c.CourseName
    FROM Exams ex
    INNER JOIN Courses c ON ex.CourseID = c.CourseID
    WHERE ExamID = @ExamId

    -- Retrieve Grade
    SELECT @Grade = sg.Grade
    FROM StudentGrades sg
    WHERE sg.CourseID = (SELECT CourseID FROM Exams WHERE ExamID = @ExamId)
    AND sg.StudentID = @StdId

    -- Return Course Name and Grade
    SELECT @CourseName AS CourseName, @Grade AS Grade
END
```

Usage

This stored procedure retrieves exam information including course name and grade.

Input Parameters

- `@ExamId`: ID of the exam (int).
- `@StdId`: ID of the student (int).

Execution Command

```
EXEC [dbo].[GetExamInfo] @ExamId = 1, @StdId = 1
```

GetExamsDeptCourse

```
CREATE PROCEDURE [dbo].[GetExamsDeptCourse]
    @crsid int,
    @deptid int
AS
BEGIN
    SELECT e.ExamID , e.ExamDate , e.Duration
    FROM Exams e
    WHERE e.CourseID = @crsid AND e.DeptID = @deptid
END
```

Usage

This stored procedure retrieves exams for a specific department and course.

Input Parameters

- `@crsid`: Course ID (int).
- `@deptid`: Department ID (int).

Execution Command

```
EXEC [dbo].[GetExamsDeptCourse] @crsid = 1, @deptid = 1
```

GetExamTable

```
CREATE PROCEDURE [dbo].[GetExamTable]
AS
BEGIN
    SELECT CourseName, ExamDate
    FROM Courses
    INNER JOIN Exams ON Courses.CourseID = Exams.CourseID
END
```

Usage

This stored procedure retrieves a table containing course names and their respective exam dates.

Execution Command

```
EXEC [dbo].[GetExamTable]
```

GetGradesByStudentID

```
CREATE PROCEDURE [dbo].[GetGradesByStudentID]
    @StudentID INT
AS
BEGIN
    SELECT Courses.CourseName, StudentGrades.Grade
    FROM Courses
    INNER JOIN StudentGrades ON Courses.CourseID = StudentGrades.CourseID
    WHERE StudentGrades.StudentID = @StudentID
END;
```

Usage

This stored procedure retrieves grades for a student based on their ID.

Input Parameters

- `@StudentID` : ID of the student (int).

Execution Command

```
EXEC [dbo].[GetGradesByStudentID] @StudentID = 1
```

GetInstructorByid

```
CREATE PROCEDURE [dbo].[GetInstructorByid]
    @id int
AS
BEGIN
    SELECT u.UserName, b.BranchName
    FROM Instructors I
    JOIN Users u ON u.UserID = I.InstructorID
    JOIN Branches b ON b.BranchID = I.BranchID
    WHERE I.InstructorID = @id
END
```

Usage

This stored procedure retrieves instructor information by their ID.

Input Parameters

- `@id`: Instructor ID (int).

Execution Command

```
EXEC [dbo].[GetInstructorByid] @id = 1
```

GetInstructorsByBranchID

```
CREATE PROCEDURE [dbo].[GetInstructorsByBranchID]
    @BranchID INT
AS
BEGIN
    SELECT Users.*
    FROM Instructors
    INNER JOIN Users ON Instructors.InstructorID = Users.UserID
    WHERE Instructors.BranchID = @BranchID;
END
```

Usage

This stored procedure retrieves instructors by branch ID.

Input Parameters

- `@BranchID`: ID of the branch (int).

Execution Command

```
EXEC [dbo].[GetInstructorsByBranchID] @BranchID = 1
```

---

getInstructorsOfDepartments

```
CREATE PROCEDURE [dbo].[getInstructorsOfDepartments]
    @deptID int
AS
BEGIN
    SELECT u.*
    FROM Branches b
    INNER JOIN Departments d ON b.BranchID = d.BranchID
    INNER JOIN Instructors i ON i.BranchID = b.BranchID
    INNER JOIN Users u ON u.UserID = i.InstructorID
    WHERE d.DeptID = @deptID
END
```

Usage

This stored procedure retrieves instructors of departments based on department ID.

Input Parameters

- `@deptID`: Department ID (int).

Execution Command

```
EXEC [dbo].[getInstructorsOfDepartments] @deptID = 1
```

GetQuestionChoices

```
CREATE PROCEDURE [dbo].[GetQuestionChoices]
    @ExamID INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        EQ.QuestionID,
        QP.Title,
        Choices = STRING_AGG(QC.Choice, ', ')
    FROM
        ExamQuestions EQ
    INNER JOIN
        QuestionPools QP ON EQ.QuestionID = QP.QuestionID
    INNER JOIN
        QuestionChoices QC ON EQ.QuestionID = QC.QuestionID
    WHERE EQ.ExamID = @ExamID
    GROUP BY
        EQ.QuestionID,
        QP.Title;
END
```

Usage

This stored procedure retrieves question choices for a specific exam.

Input Parameters

- `@ExamID`: ID of the exam (int).

Execution Command

```
EXEC [dbo].[GetQuestionChoices] @ExamID = 1
```

GetRandomQuestions

```
CREATE PROCEDURE [dbo].[GetRandomQuestions]
    @mcqCount INT,
    @TFcount INT,
    @courseId INT,
    @examId INT
AS
BEGIN
    INSERT INTO ExamQuestions
    SELECT TOP (@TFcount) @examId , QuestionID
    FROM QuestionPools
    WHERE QuestionType = 0 AND CourseID = @courseId
    ORDER BY NEWID();

    INSERT INTO ExamQuestions
    SELECT TOP (@mcqCount) @examId , QuestionID
    FROM QuestionPools
    WHERE QuestionType = 1 AND CourseID = @courseId
    ORDER BY NEWID();
END;
```

Usage

This stored procedure selects random questions for an exam based on the count of multiple-choice and true/false questions.

Input Parameters

- `@mcqCount`: Number of multiple-choice questions (int).
- `@TFcount`: Number of true/false questions (int).
- `@courseId`: ID of the course (int).

- `@examId`: ID of the exam (int).

Execution Command

```
EXEC [dbo].[GetRandomQuestions] @mcqCount = 5, @TFcount = 5, @courseId = 1, @examId
= 1
```

GetStudentExamDetails

```
CREATE PROCEDURE [dbo].[GetStudentExamDetails]
    @StudentID INT,
    @ExamID INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        QP.Title AS [Question Title],
        SEQ.SelectedAnswerIndex,
        QP.CorrectAnswerIndex,
        QP.Grade AS [Question Grade]
    FROM
        StudentExamQuestions SEQ
    INNER JOIN
        QuestionPools QP ON SEQ.QuestionID = QP.QuestionID
    WHERE
        SEQ.StudentID = @StudentID
        AND SEQ.ExamID = @ExamID;
END
```

Usage

This stored procedure retrieves details of a student's exam, including the selected answer index, correct answer index, and question grade.

Input Parameters

- `@StudentID`: ID of the student (int).

- `@ExamID`: ID of the exam (int).

Execution Command

```
EXEC [dbo].[GetStudentExamDetails] @StudentID = 1, @ExamID = 1
```

GetStudentGrades

```
CREATE PROCEDURE [dbo].[GetStudentGrades]
    @StudentID INT,
    @ExamID INT,
        @CourseID INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @TotalGrade INT;
        DECLARE @StudentGrades INT;

    -- Fetch the student exam details using the GetStudentExamDetails stored
procedure
    DECLARE @StudentExamDetails TABLE (
        [Question Title] NVARCHAR(200),
        [SelectedAnswerIndex] INT,
        [CorrectAnswerIndex] INT,
        [Question Grade] INT
    );

    INSERT INTO @StudentExamDetails EXEC [dbo].[GetStudentExamDetails] @StudentID,
@ExamID;

    SELECT @StudentGrades = SUM(CASE WHEN SelectedAnswerIndex = CorrectAnswerIndex
THEN [Question Grade] ELSE 0 END)
          , @TotalGrade = SUM([Question Grade])
    FROM @StudentExamDetails;
    INSERT INTO [iti_exam].[dbo].[StudentGrades] ([CourseID], [StudentID], [Grade])
    VALUES (@CourseID, @StudentID, @StudentGrades);
    SELECT @StudentGrades AS [Student Grade], @TotalGrade AS [Total Grade] ;
END
```

Usage

This stored procedure calculates and inserts the student's grade for an exam into the database.

Input Parameters

- `@StudentID`: ID of the student (int).
- `@ExamID`: ID of the exam (int).
- `@CourseID`: ID of the course (int).

Execution Command

```
EXEC [dbo].[GetStudentGrades] @StudentID = 1, @ExamID = 1, @CourseID = 1
```

GetStudentName

```
CREATE PROCEDURE [dbo].[GetStudentName]
    @studentID INT
AS
BEGIN
    SELECT UserName
    FROM Users
    WHERE UserID = @studentID;
END;
```

Usage

This stored procedure retrieves the name of a student based on their ID.

Input Parameters

- `@studentID`: ID of the student (int).

Execution Command

```
EXEC [dbo].[GetStudentName] @studentID = 1
```

GetStudentsByDept

```
CREATE PROCEDURE [dbo].[GetStudentsByDept]
    @deptid int
AS
BEGIN
    SELECT u.UserID, u.UserName, u.Email
    FROM Users u
    JOIN Students s ON u.UserID = s.StudentID
    WHERE s.DeptID = @deptid
END
```

Usage

This stored procedure retrieves students belonging to a specific department.

Input Parameters

- `@deptid`: Department ID (int).

Execution Command

```
EXEC [dbo].[GetStudentsByDept] @deptid = 1
```

GetStudentsByDeptID

```
CREATE PROCEDURE [dbo].[GetStudentsByDeptID]
    @DeptID INT
AS
```

```sql
BEGIN
    SELECT Users.*
    FROM Students
    INNER JOIN Users ON Students.StudentID = Users.UserID
    WHERE Students.DeptID = @DeptID;
END
```

Usage

This stored procedure retrieves students belonging to a specific department by department ID.

Input Parameters

- `@DeptID`: ID of the department (int).

Execution Command

```sql
EXEC [dbo].[GetStudentsByDeptID] @DeptID = 1
```

---

getStudentsPerCourse

```sql
CREATE PROCEDURE [dbo].[getStudentsPerCourse]
    @insid int
AS
BEGIN
    SELECT COUNT(StudentID) AS [Number of students], c.CourseName
    FROM Students s
    JOIN Departments d ON d.DeptID = s.DeptID
    JOIN Branches b ON b.BranchID = d.BranchID
    JOIN Instructors I ON I.BranchID = b.BranchID
    JOIN Instructor_Courses ic ON ic.InstructorID = i.InstructorID
    JOIN Courses c ON c.CourseID = ic.CourseID
    WHERE I.InstructorID = @insid
    GROUP BY c.CourseName
END
```

Usage

This stored procedure retrieves the number of students enrolled in each course taught by a specific instructor.

Input Parameters

- `@insid`: ID of the instructor (int).

Execution Command

```sql
EXEC [dbo].[getStudentsPerCourse] @insid = 1
```

---

getTopicByNameAndCourseID

```sql
CREATE PROCEDURE [dbo].[getTopicByNameAndCourseID]
    @courseId int,
    @topicName nvarchar(255)
AS
BEGIN
    SELECT * FROM CourseTopics
    WHERE CourseID = @courseId AND Topic = @topicName
END
```

Usage

This stored procedure retrieves a topic based on its name and the ID of the course it belongs to.

Input Parameters

- `@courseId`: ID of the course (int).
- `@topicName`: Name of the topic (nvarchar(255)).

Execution Command

```
EXEC [dbo].[getTopicByNameAndCourseID] @courseId = 1, @topicName = 'Topic Name'
```

InitializeBranches

```
CREATE PROCEDURE [dbo].[InitializeBranches]
AS
BEGIN
    INSERT INTO Branches (BranchName)
    VALUES ('Mansoura'), ('Smart Village'), ('New Capital');
END
```

This stored procedure initializes branches by inserting their names into the `Branches` table.

InitializeCourses

```
CREATE PROCEDURE [dbo].[InitializeCourses]
AS
BEGIN
    INSERT INTO Courses (CourseName)
    VALUES ('Python'), ('Java'), ('C++'), ('JavaScript'), ('HTML and CSS');
END
```

This stored procedure initializes courses by inserting their names into the `Courses` table.

InitializeDepartments

```
CREATE PROCEDURE [dbo].[InitializeDepartments]
AS
BEGIN
    INSERT INTO Departments (DeptName, BranchID)
    VALUES ('PD', 1), ('AI', 1), ('OS', 1);
END
```

This stored procedure initializes departments by associating them with branches.

InitializeExams

```
CREATE PROCEDURE [dbo].[InitializeExams]
AS
BEGIN
    INSERT INTO Exams (ExamDate, Duration, CourseID, DeptID)
    VALUES ('2024-03-21', 120, 4, 1);
END
```

This stored procedure initializes exams by inserting exam information into the `Exams` table.

InitializeInstructors

```
CREATE PROCEDURE [dbo].[InitializeInstructors]
AS
BEGIN
    INSERT INTO Instructors (BranchID, InstructorID)
    VALUES (1, 4), (1, 5);
END
```

This stored procedure initializes instructors by assigning them to specific branches.

InitializeQuestionChoices

```
Create PROCEDURE [dbo].[InitializeQuestionChoices]
AS
BEGIN
   INSERT INTO QuestionChoices (Choice, QuestionID, ChoiceIndex)

    VALUES ('JavaScript is a scripting language.', 0, 0),

           ('JavaScript is a markup language.', 0, 1),

           ('JavaScript is a programming language.', 0, 2),
           ('JavaScript is a styling language.', 0, 3);
   INSERT INTO QuestionChoices (Choice, QuestionID, ChoiceIndex)

    VALUES ('Number', 1, 0),

           ('String', 1, 1),

           ('Boolean', 1, 2),

           ('All of the above', 1, 3);
          -- etc

END
```

This stored procedure initializes question choices for exam questions.

InitializeQuestionPools

```
CREATE PROCEDURE [dbo].[InitializeQuestionPools]
AS
BEGIN
      INSERT INTO QuestionPools (Title, QuestionType, CorrectAnswerIndex, Grade,
CourseID)

    VALUES

        ('What is JavaScript?', 1, 0, 10, 4),

        ('What are the data types in JavaScript?', 1, 2, 10, 4),

        ('What is the purpose of the "use strict" directive in JavaScript?', 1, 1,
10, 4),

        ('What is event bubbling in JavaScript?', 1, 3, 10, 4)
        -- etc
END
```

This stored procedure initializes question pools for exams.

InitializeStudentExamQuestions

```
CREATE PROCEDURE [dbo].[InitializeStudentExamQuestions]
AS
BEGIN
    INSERT INTO StudentExamQuestions (SelectedAnswerIndex, ExamID, QuestionID,
StudentID)

    VALUES (1, 0, 0, 1),

           (2, 0, 1, 1),

           (1, 0, 2, 1),

           (2, 0, 3, 1);
END
```

This stored procedure initializes student exam questions.

InitializeStudents

```
CREATE PROCEDURE [dbo].[InitializeStudents]
AS
BEGIN
    INSERT INTO Students (DeptID, StudentID)
    VALUES (1, 1), (1, 2);
END
```

This stored procedure initializes students by associating them with departments.

InitializeUsers

```
CREATE PROCEDURE [dbo].[InitializeUsers]
AS
BEGIN
   INSERT INTO Users (UserName, Email, Address, Password, Phone, Gender)

    VALUES ('Ahmed', 'ahmed@example.com', 'Address1', '123', '1234567890', 'M'),
           ('Maryam', 'maryam@example.com', 'Address2', '123', '0987654321', 'F'),

           ('Omar', 'omar@example.com', 'Address3', '123', '9876543210', 'M'),

           ('Abdo', 'abdo@example.com', 'Address4', '123', '0123456789', 'M'),

           ('Saber', 'saber@example.com', 'Address5', '123', '5678901234', 'M');
END
```

This stored procedure initializes user information.

InsertOrUpdateSelectedAnswerIndex

```
CREATE PROCEDURE [dbo].[InsertOrUpdateSelectedAnswerIndex]
    @StudentID INT,
    @ExamID INT,
    @QuestionID INT,
    @SelectedAnswerIndex INT
AS
BEGIN
    BEGIN TRANSACTION;

    -- Check if the record already exists
    IF EXISTS (SELECT 1 FROM StudentExamQuestions WHERE StudentID = @StudentID AND
ExamID = @ExamID AND QuestionID = @QuestionID)
    BEGIN
        -- Update the existing record
        UPDATE StudentExamQuestions
        SET SelectedAnswerIndex = @SelectedAnswerIndex
        WHERE StudentID = @StudentID AND ExamID = @ExamID AND QuestionID =
@QuestionID;
    END
    ELSE
    BEGIN
        -- Insert a new record
        INSERT INTO StudentExamQuestions (StudentID, ExamID, QuestionID,
SelectedAnswerIndex)
        VALUES (@StudentID, @ExamID, @QuestionID, @SelectedAnswerIndex);
    END

    COMMIT TRANSACTION;
END
```

Usage

This stored procedure inserts or updates the selected answer index for a student in an exam.

Input Parameters

- `@StudentID` (INT): ID of the student.
- `@ExamID` (INT): ID of the exam.
- `@QuestionID` (INT): ID of the question.
- `@SelectedAnswerIndex` (INT): Index of the selected answer.

Execution Command

```
EXEC [dbo].[InsertOrUpdateSelectedAnswerIndex] @StudentID = 1, @ExamID = 1,
@QuestionID = 1, @SelectedAnswerIndex = 2
```

InsertSelectedAnswerIndex

```
CREATE PROCEDURE [dbo].[InsertSelectedAnswerIndex]
    @StudentID INT,
    @ExamID INT,
    @QuestionID INT,
    @SelectedAnswerIndex INT
AS
BEGIN
    INSERT INTO StudentExamQuestions (SelectedAnswerIndex, ExamID, QuestionID,
StudentID)
    VALUES (@SelectedAnswerIndex, @ExamID, @QuestionID, @StudentID);
END
```

Usage

This stored procedure inserts the selected answer index for a student in an exam.

Input Parameters

- `@StudentID` (INT): ID of the student.
- `@ExamID` (INT): ID of the exam.
- `@QuestionID` (INT): ID of the question.
- `@SelectedAnswerIndex` (INT): Index of the selected answer.

Execution Command

```
EXEC [dbo].[InsertSelectedAnswerIndex] @StudentID = 1, @ExamID = 1, @QuestionID =
1, @SelectedAnswerIndex = 2
```

RemoveInstructorCourses

```
CREATE PROCEDURE [dbo].[RemoveInstructorCourses]
    @InstructorID INT
AS
BEGIN
    -- Delete the entry
    DELETE FROM Instructor_Courses WHERE  InstructorID = @InstructorID
END
```

Usage

This stored procedure removes courses assigned to an instructor.

Input Parameters

- `@InstructorID` (INT): ID of the instructor.

Execution Command

```
EXEC [dbo].[RemoveInstructorCourses] @InstructorID = 1
```

RemoveInstructorFromCourse

```
CREATE PROCEDURE [dbo].[RemoveInstructorFromCourse]
    @CourseID INT,
    @InstructorID INT
AS
BEGIN
    -- Delete the entry
    DELETE FROM Instructor_Courses WHERE CourseID = @CourseID AND InstructorID =
@InstructorID
END
```

Usage

This stored procedure removes an instructor from a specific course.

Input Parameters

- `@CourseID` (INT): ID of the course.
- `@InstructorID` (INT): ID of the instructor.

Execution Command

```
EXEC [dbo].[RemoveInstructorFromCourse] @CourseID = 1, @InstructorID = 1
```

Reseed

```
CREATE PROCEDURE [dbo].[Reseed]
AS
BEGIN
    EXEC sp_MSforeachtable 'DBCC CHECKIDENT (''?'', RESEED, 0)'
END
```

Usage

This stored procedure resets the identity seed for all tables in the database.

Input Parameters

None

Execution Command

```
EXEC [dbo].[Reseed]
```

---

updateBranchName

```
create proc [dbo].[updateBranchName] @id int, @newName nvarchar(255)
as
begin
        update Branches set BranchName = @newName where BranchID=@id
end
```

Usage

This stored procedure updates the name of a branch.

Input Parameters

- `@id` (INT): ID of the branch.
- `@newName` (NVARCHAR(255)): New name for the branch.

Execution Command

```
EXEC [dbo].[updateBranchName] @id = 1, @newName = 'New Branch Name'
```

UpdateDepartment

```
CREATE PROCEDURE [dbo].[UpdateDepartment]
    @deptId int,
    @deptName nvarchar(255),
    @BranchID int
AS
BEGIN
    UPDATE Departments SET DeptName = @deptName, BranchID = @BranchID WHERE DeptID
= @deptId
END
```

Usage

This stored procedure updates the name and branch of a department.

Input Parameters

- `@deptId` (INT): ID of the department.
- `@deptName` (NVARCHAR(255)): New name for the department.
- `@BranchID` (INT): ID of the branch for the department.

Execution Command

```
EXEC [dbo].[UpdateDepartment] @deptId = 1, @deptName = 'New Department Name',
@BranchID = 1
```

UpdateInstructor

```
CREATE PROCEDURE [dbo].[UpdateInstructor]
    @UserID INT,
    @UserName NVARCHAR(50),
    @Email NVARCHAR(50),
    @Address NVARCHAR(100),
    @Password NVARCHAR(50),
    @Phone NVARCHAR(20),
    @Gender CHAR(1),
```

```sql
    @BranchID INT
AS
BEGIN
    UPDATE Users
    SET UserName = @UserName,
        Email = @Email,
        Address = @Address,
        Password = @Password,
        Phone = @Phone,
        Gender = @Gender
    WHERE UserID = @UserID;

    UPDATE Instructors
    SET BranchID = @BranchID
    WHERE InstructorID = @UserID;
END
```

Usage

This stored procedure updates the information of an instructor in the database.

Input Parameters

- `@UserID` (INT): ID of the instructor.
- `@UserName` (NVARCHAR(50)): New username for the instructor.
- `@Email` (NVARCHAR(50)): New email address for the instructor.
- `@Address` (NVARCHAR(100)): New address for the instructor.
- `@Password` (NVARCHAR(50)): New password for the instructor.
- `@Phone` (NVARCHAR(20)): New phone number for the instructor.
- `@Gender` (CHAR(1)): New gender for the instructor ('M' for male, 'F' for female).
- `@BranchID` (INT): ID of the branch to which the instructor belongs.

Execution Command

```sql
EXEC [dbo].[UpdateInstructor] @UserID = 1, @UserName = 'NewUsername', @Email =
'newemail@example.com', @Address = 'New Address', @Password = 'newpassword', @Phone
= '1234567890', @Gender = 'M', @BranchID = 1
```

UpdateStudent

```sql
CREATE PROCEDURE [dbo].[UpdateStudent]
    @UserID INT,
    @UserName NVARCHAR(50),
    @Email NVARCHAR(50),
    @Address NVARCHAR(100),
    @Password NVARCHAR(50),
    @Phone NVARCHAR(20),
    @Gender CHAR(1),
    @DeptID INT
AS
BEGIN
    UPDATE Users
    SET UserName = @UserName,
        Email = @Email,
        Address = @Address,
        Password = @Password,
        Phone = @Phone,
        Gender = @Gender
    WHERE UserID = @UserID;

    UPDATE Students
    SET DeptID = @DeptID
    WHERE StudentID = @UserID;
END
```

Usage

This stored procedure updates the information of a student in the database.

Input Parameters

- `@UserID` (INT): ID of the student.
- `@UserName` (NVARCHAR(50)): New username for the student.
- `@Email` (NVARCHAR(50)): New email address for the student.
- `@Address` (NVARCHAR(100)): New address for the student.
- `@Password` (NVARCHAR(50)): New password for the student.
- `@Phone` (NVARCHAR(20)): New phone number for the student.
- `@Gender` (CHAR(1)): New gender for the student ('M' for male, 'F' for female).
- `@DeptID` (INT): ID of the department to which the student belongs.

Execution Command

```
EXEC [dbo].[UpdateStudent] @UserID = 1, @UserName = 'NewUsername', @Email =
'newemail@example.com', @Address = 'New Address', @Password = 'newpassword', @Phone
= '1234567890', @Gender = 'M', @DeptID = 1
```

---

getStudentsInfobydeptid

```
CREATE PROCEDURE [dbo].[getStudentsInfobydeptid]
    @DeptID INT
AS
BEGIN
    SELECT Users.UserID, Users.UserName, Users.Email, Users.Address,
Branches.BranchName, Departments.DeptName, Users.Phone, Users.Gender
    FROM Branches
    INNER JOIN Departments ON Branches.BranchID = Departments.BranchID
    INNER JOIN Students ON Departments.DeptID = Students.DeptID
    INNER JOIN Users ON Students.StudentID = Users.UserID
    WHERE Departments.DeptID = @DeptID;
END;
```

Usage

This stored procedure retrieves information about students belonging to a specific department.

Input Parameters

- `@DeptID` (INT): ID of the department.

Execution Command

```
EXEC [dbo].[getStudentsInfobydeptid] @DeptID = 1
```

GetAllDeptCoursesforIns

```sql
create proc [dbo].[GetAllDeptCoursesforIns] @deptid int , @insid int
as
begin
select c.CourseID , c.CourseName from Courses c join Course_Dept dc on
dc.CourseID = c.CourseID
join Departments d on d.DeptID = dc.DeptId
join Instructor_Courses ic on ic.CourseID = c.CourseID
where d.DeptID = @deptid and ic.InstructorID = @insid
end
```

Usage

This stored procedure retrieves all courses belonging to a specific department for a given instructor.

Input Parameters

- `@deptid` (INT): ID of the department.
- `@insid` (INT): ID of the instructor.

Execution Command

```sql
EXEC [dbo].[GetAllDeptCoursesforIns] @deptid = 1, @insid = 1
```