LEARNIFY

Онлайн платформа за курсове

1. Резултати от създадените функции, приложени върху данните от база данни LearnifyDB:

Функция GET_COURSE_AVERAGE_RATING

- **Описание:** Връща средния рейтинг на курс по неговото ID.

```
CREATE FUNCTION dbo.GetAverageRating(@CourseID INT)
RETURNS FLOAT
AS
BEGIN
    DECLARE @Avg FLOAT;

    SELECT @Avg = AVG(CAST(Rating AS FLOAT))
    FROM Reviews
    WHERE CourseID = @CourseID;

    RETURN @Avg;
END;

-- 1. Средна оценка на курс
-- Тествам за курс Основи на C#
DECLARE @CSharpID INT;
SELECT @CSharpID = CourseID FROM Courses WHERE Title = 'Основи на C#';

SELECT dbo.GetAverageRating(@CSharpID) AS AvgRating;
```

Функция: IsEmailUnique

Описание:
Проверява дали даден имейл съществува в таблицата Users.
Връща:

- **1** – ако имейлът *не съществува* (уникален)
- **0** – ако имейлът *вече е зает*

Приложение върху данни от база:

```
CREATE FUNCTION dbo.IsEmailUnique(@Email NVARCHAR(120))
RETURNS BIT
AS
BEGIN
    DECLARE @Exists INT;

    SELECT @Exists = COUNT(*)
    FROM Users
    WHERE Email = @Email;

    RETURN CASE WHEN @Exists = 0 THEN 1 ELSE 0 END;
END;
GO
```

-- 2. Проверка за уникален имейл

```
SELECT dbo.IsEmailUnique('ivan.ivanov@example.com') AS IsEmailUnique; -- очаква 0
SELECT dbo.IsEmailUnique('newstudent@example.com') AS IsEmailUnique;   -- очаква 1
```

Функция: CountStudentsInCourse

Описание:
Връща броя записани студенти в даден курс по неговото `CourseID`.

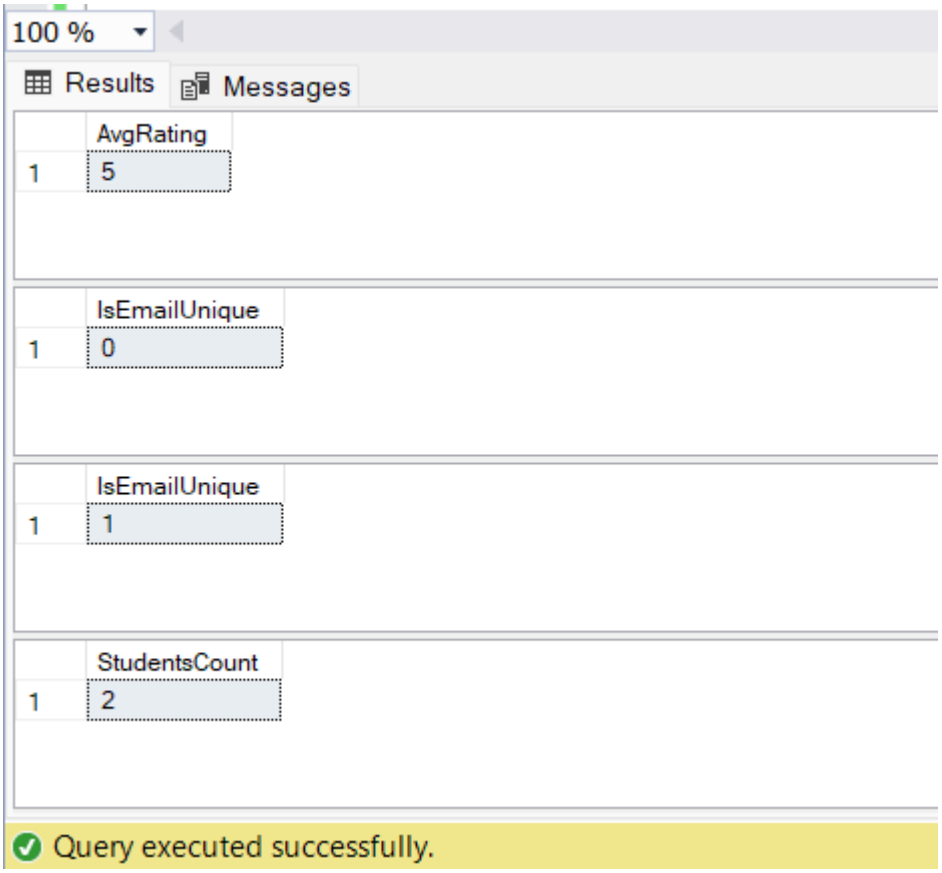
Приложение върху данни от база:

```
-- Брой студенти записани на курс


CREATE FUNCTION dbo.CountStudentsInCourse(@CourseID INT)

RETURNS INT

AS

BEGIN

    DECLARE @Count INT;


    SELECT @Count = COUNT(*)

    FROM Enrollments

    WHERE CourseID = @CourseID;


    RETURN @Count;

END;

GO
```

-- 3. Брой студенти в курс

```
SELECT dbo.CountStudentsInCourse(@CSharpID) AS StudentsCount;
```

Резултат:



2. Резултати от създадените съхранени процедури, приложени върху данните от база данни LearnifyDB:

Процедура: AddCourse

Описание:
Добавя нов курс в таблицата Courses.

Параметри:

- @Title NVARCHAR(150) – заглавие на курса
- @CategoryID INT – ID на категорията
- @InstructorID INT – ID на инструктора
- @Description NVARCHAR(MAX) – описание на курса

Приложение върху данни от база:

```
-- Добавяне на нов курс
CREATE PROCEDURE AddCourse
    @Title NVARCHAR(150),
    @CategoryID INT,
    @InstructorID INT,
    @Description NVARCHAR(MAX)
AS
BEGIN
    INSERT INTO Courses (Title, CategoryID, InstructorID, Description)
    VALUES (@Title, @CategoryID, @InstructorID, @Description);
END;

--  Тест на stored procedures
-- a) AddCourse
DECLARE @NewCourseID INT;
EXEC AddCourse
    @Title='Тест курс',
    @CategoryID=@ProgID,
    @InstructorID=@GeorgiID,
    @Description='Тест на процедурата AddCourse';
SELECT @NewCourseID = CourseID FROM Courses WHERE Title='Тест курс';
```

Процедура: EnrollStudent

Описание:
Записва студент за даден курс.

Параметри:

- *@StudentID INT – ID на студента*
- *@CourseID INT – ID на курса*

Приложение върху данни от база:

```
-- Записване на студент за курс
CREATE PROCEDURE EnrollStudent
   @StudentID INT,
   @CourseID INT
AS
BEGIN
   INSERT INTO Enrollments (StudentID, CourseID, EnrolledAt)
   VALUES (@StudentID, @CourseID, GETDATE());
END;
GO
```

-- 2. EnrollStudent тестване

```
IF @NewCourseID IS NOT NULL
BEGIN
   EXEC EnrollStudent
      @StudentID=@IvanID,
      @CourseID=@NewCourseID;

   -- Проверка на записването
   SELECT * FROM Enrollments WHERE CourseID=@NewCourseID AND
StudentID=@IvanID;
END
ELSE
BEGIN
```

PRINT 'Грешка: @NewCourseID е NULL, записването не може да се извърши.';

END

Процедура: GetCoursesByCategory

Описание:
Връща всички курсове от дадена категория.

Параметри:

- *`@CategoryID INT` – ID на категорията*

Приложение върху данни от база:

-- Връща всички курсове от дадена категория

CREATE PROCEDURE GetCoursesByCategory

@CategoryID INT

AS

BEGIN

SELECT * FROM Courses WHERE CategoryID = @CategoryID;

END;

-- 3. GetCoursesByCategory Тестване

DECLARE @CategoryID INT;

-- Взимаме ID на категория "Програмиране"

SELECT @CategoryID = CategoryID

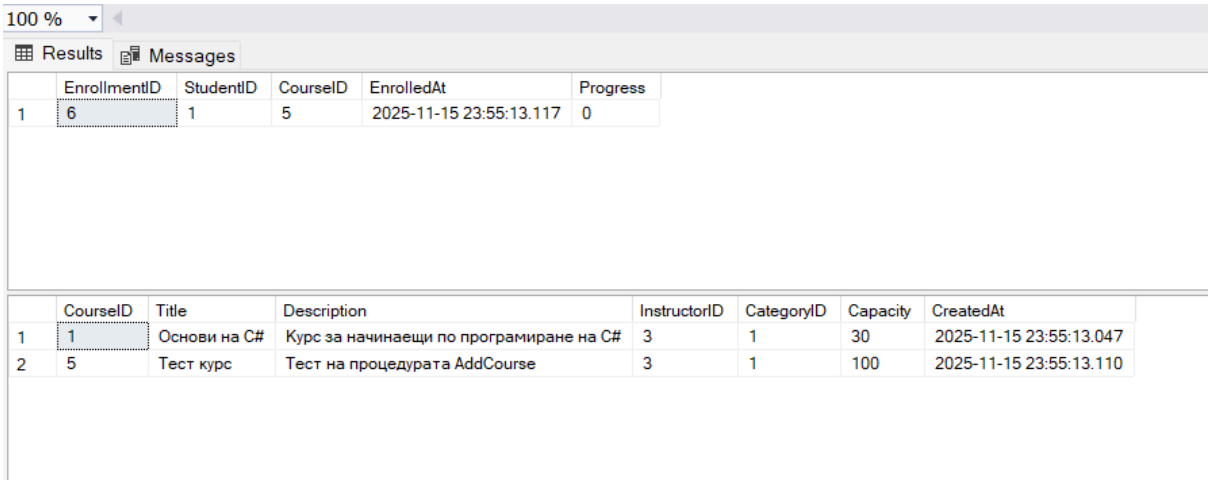
FROM Categories

WHERE Name = 'Програмиране';

-- Извикваме процедурата

EXEC GetCoursesByCategory @CategoryID = @CategoryID;

Резултат:



100 %

Results Messages

	EnrollmentID	StudentID	CourseID	EnrolledAt	Progress
1	6	1	5	2025-11-15 23:55:13.117	0

	CourseID	Title	Description	InstructorID	CategoryID	Capacity	CreatedAt
1	1	Основи на C#	Курс за начинаещи по програмиране на C#	3	1	30	2025-11-15 23:55:13.047
2	5	Тест курс	Тест на процедурата AddCourse	3	1	100	2025-11-15 23:55:13.110

3. Резултати от създадените тригери, приложени върху данните от база данни LearnifyDB:

Trigger: SetDefaultRole

Описание:
Автоматично задава ролята Student за нов потребител, ако не е зададена.

Приложение върху данни от база:

```
-- Автоматично задаване на ролята на нов потребител

CREATE TRIGGER SetDefaultRole

ON Users

AFTER INSERT

AS

BEGIN

    UPDATE Users

    SET Role = 'Student'

    WHERE Role IS NULL AND UserID IN (SELECT UserID FROM inserted);

END;
```

-- TEST 1: SetDefaultRole

INSERT INTO Users (FullName, Email, PasswordHash, Role)

VALUES ('TR User', 'tr_user_role_001@example.com', 'h', NULL);

*SELECT * FROM Users WHERE Email='tr_user_role_001@example.com';*

Trigger: CheckCourseCapacity

Описание:
Не позволява записване на студенти в курс, ако капацитетът е надвишен.

Приложение върху данни от база:

-- Ограничение на капацитета на курса

CREATE TRIGGER CheckCourseCapacity

ON Enrollments

AFTER INSERT

AS

BEGIN

-- Проверка дали броят записани студенти за даден курс надвишава капацитета

IF EXISTS (

SELECT 1

FROM Enrollments e

JOIN Courses c ON e.CourseID = c.CourseID

GROUP BY e.CourseID, c.Capacity

HAVING COUNT(*) > c.Capacity

```
    )
    BEGIN
        ROLLBACK; -- Отказва записването
        RAISERROR('Курсът е запълнен. Няма места.',16,1);
    END
END;
```

-- TEST 2: CheckCourseCapacity

```
INSERT INTO Users (FullName, Email, PasswordHash, Role)
VALUES ('TR Instructor', 'tr_instr_001@example.com', 'h', 'Instructor');
DECLARE @I INT = SCOPE_IDENTITY();

INSERT INTO Categories (Name, Description)
VALUES ('TR_Cat_001','d');
DECLARE @C INT = SCOPE_IDENTITY();

INSERT INTO Courses (Title, CategoryID, InstructorID, Description, Capacity)
VALUES ('TR_Course_001', @C, @I, 'd', 1);
DECLARE @Course INT = SCOPE_IDENTITY();

INSERT INTO Users (FullName, Email, PasswordHash, Role)
VALUES ('TR Student1', 'tr_st1_001@example.com', 'h', 'Student');
DECLARE @S1 INT = SCOPE_IDENTITY();
```

INSERT INTO Enrollments (StudentID, CourseID)

VALUES (@S1, @Course);

INSERT INTO Users (FullName, Email, PasswordHash, Role)

VALUES ('TR Student2', 'tr_st2_001@example.com', 'h', 'Student');

DECLARE @S2 INT = SCOPE_IDENTITY();

BEGIN TRY

INSERT INTO Enrollments (StudentID, CourseID)

VALUES (@S2, @Course);

END TRY

BEGIN CATCH

PRINT ERROR_MESSAGE();

END CATCH;

*SELECT * FROM Enrollments WHERE CourseID = @Course;*

Trigger: UpdateProgressOnReview

Описание:
При добавяне на ревю, автоматично се задава Progress = 100 за студента в съответния курс.

Приложение върху данни от база:

-- Автоматично обновяване на прогрес при добавяне на ревю

CREATE TRIGGER UpdateProgressOnReview

ON Reviews

```
AFTER INSERT
AS
BEGIN
    UPDATE Enrollments
    SET Progress = 100
    WHERE CourseID IN (SELECT CourseID FROM inserted)
      AND StudentID IN (SELECT StudentID FROM inserted);
END;
```

-- TEST 3: UpdateProgressOnReview

```
INSERT INTO Users (FullName, Email, PasswordHash, Role)
VALUES ('TR Student3', 'tr_st3_001@example.com', 'h', 'Student');
DECLARE @S3 INT = SCOPE_IDENTITY();

INSERT INTO Users (FullName, Email, PasswordHash, Role)
VALUES ('TR Instructor3', 'tr_instr3_001@example.com', 'h', 'Instructor');
DECLARE @I3 INT = SCOPE_IDENTITY();

INSERT INTO Categories (Name, Description)
VALUES ('TR_Cat_003','d');
DECLARE @C3 INT = SCOPE_IDENTITY();

INSERT INTO Courses (Title, CategoryID, InstructorID, Description, Capacity)
VALUES ('TR_Course_003', @C3, @I3, 'd', 5);
```

```
DECLARE @Course3 INT = SCOPE_IDENTITY();

INSERT INTO Enrollments (StudentID, CourseID, Progress)
VALUES (@S3, @Course3, 20);

INSERT INTO Reviews (StudentID, CourseID, Rating, Comment)
VALUES (@S3, @Course3, 5, 'ok');

SELECT * FROM Enrollments
WHERE StudentID = @S3 AND CourseID = @Course3;
```

Резултат:

100 %

Results | Messages

	UserID	FullName	Email	PasswordHash	Role	CreatedAt
1	32	TR User	tr_user_role_001@example.com	h	Student	2025-11-15 22:56:02.987

	EnrollmentID	StudentID	CourseID	EnrolledAt	Progress
1	21	34	12	2025-11-15 22:56:03.010	0

	EnrollmentID	StudentID	CourseID	EnrolledAt	Progress
1	23	36	13	2025-11-15 22:56:03.030	100

Query executed successfully.

```
(0 rows affected)
Курсът е запълнен. Няма места.
```