CIS-552: DATABASE DESIGN

Submitted by – Yashika Patil (02115374)

LAB – 3:

In this lab, I created a user-defined stored procedure in SQL Server that takes a student name and calculates their GPA. The GPA is computed based on the grades where the points are represented as follows: A=4, B=3, C=3, D=1, F=0.

1. **DATABASE SETUP:**
   First, I executed SQL 'CREATE TABLE' commands to create the following tables in a query.
   a. STUDENT:
      i. Name (NVARCHAR(50)): Name of the student.
      ii. Student_number (INT): Unique identifier for each student.
      iii. Class (INT): Class of the student.
      iv. Major (NVARCHAR(50)): Major of the student.

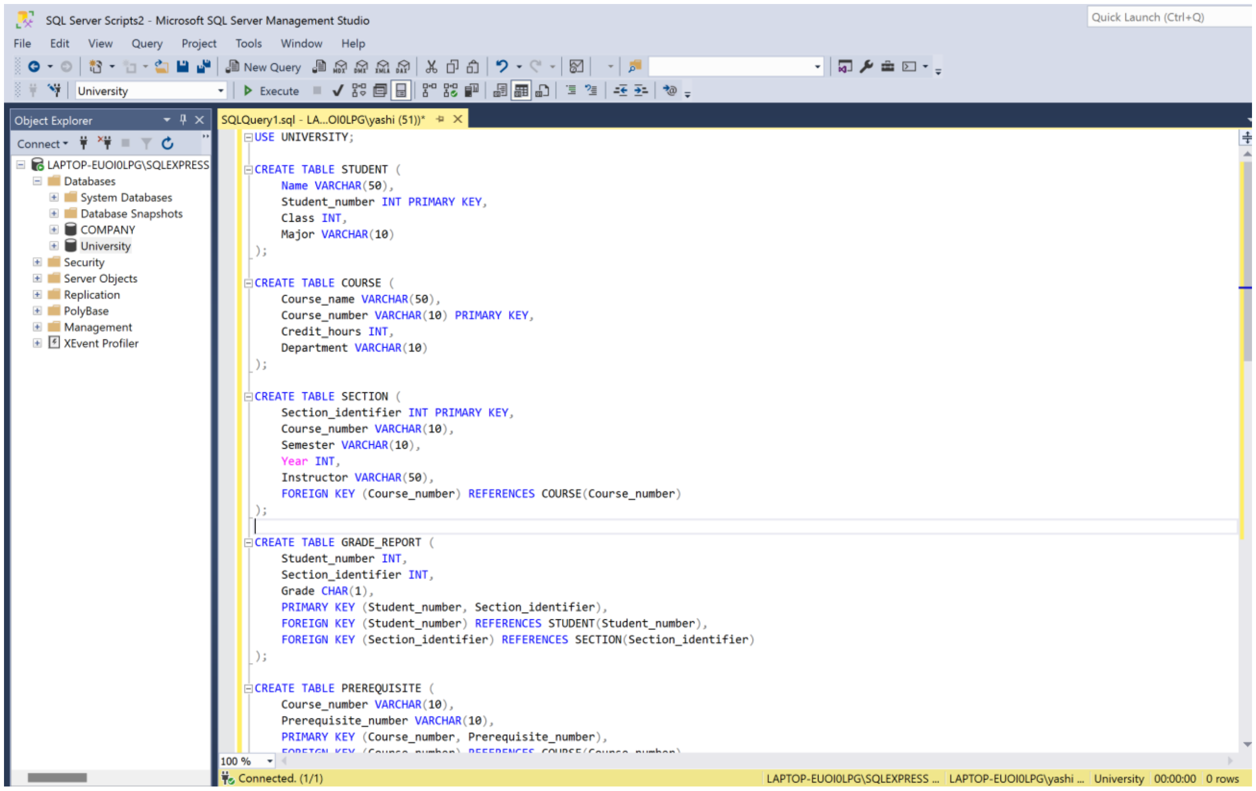
   b. COURSE:
      i. Course_name (NVARCHAR(50)): Name of the course.
      ii. Course_number (NVARCHAR(10)): Unique identifier for each course.
      iii. Credit_hours (INT): Number of credit hours for a course.
      iv. Department (NVARCHAR(50)): Department offering the course.

   c. SECTION:
      i. Section_identifier (INT): Primary key identifying each section.
      ii. Course_number (NVARCHAR(10)): Course identifier linked to the COURSE table.
      iii. Semester (NVARCHAR(10)): Semester in which the section is offered (e.g., Fall, Spring).
      iv. Year (INT): Year in which the section is offered.
      v. Instructor (NVARCHAR(50)): Name of the instructor teaching the section.
   d. GRADE_REPORT:
      i. Student_number (INT): Identifier related to the STUDENT table.
      ii. Section_identifier (INT): Identifier related to the SECTION table.
      iii. Grade (CHAR(1)): Grade of a course (A, B, C, D, F).

   e. PREREQUISITE:
      i. Course_number (NVARCHAR(10)): Course for which prerequisites exist.

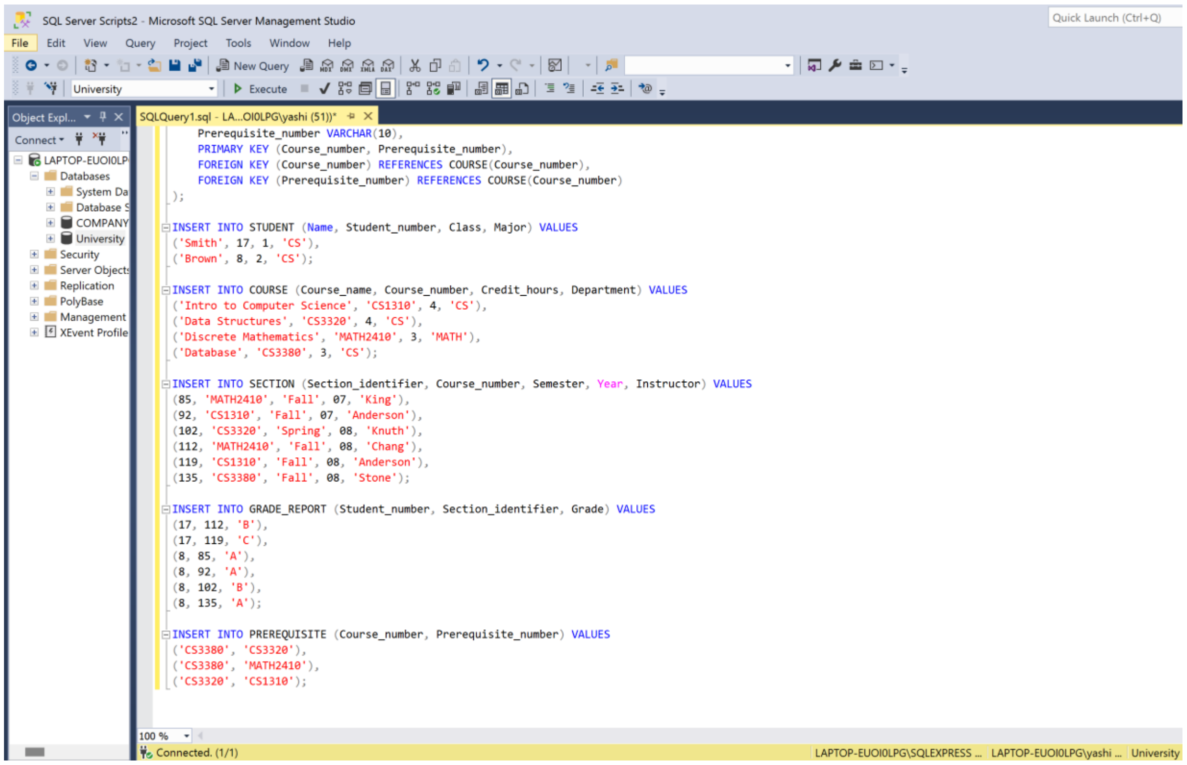
ii. Prerequisite_number (NVARCHAR(10)): Prerequisite course for a specified course.

The tables are created as shown in the screenshot below:


SQL Server Scripts2 - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
University
Execute
Object Explorer
Connect
LAPTOP-EUOI0LPG\SQLEXPRESS
Databases
System Databases
Database Snapshots
COMPANY
University
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
SQLQuery1.sql - LA...OI0LPG\yashi (51))*
100 %
Connected. (1/1)
LAPTOP-EUOI0LPG\SQLEXPRESS ... LAPTOP-EUOI0LPG\yashi ... University 00:00:00 0 rows

```sql
USE UNIVERSITY;

CREATE TABLE STUDENT (
    Name VARCHAR(50),
    Student_number INT PRIMARY KEY,
    Class INT,
    Major VARCHAR(10)
);

CREATE TABLE COURSE (
    Course_name VARCHAR(50),
    Course_number VARCHAR(10) PRIMARY KEY,
    Credit_hours INT,
    Department VARCHAR(10)
);

CREATE TABLE SECTION (
    Section_identifier INT PRIMARY KEY,
    Course_number VARCHAR(10),
    Semester VARCHAR(10),
    Year INT,
    Instructor VARCHAR(50),
    FOREIGN KEY (Course_number) REFERENCES COURSE(Course_number)
);

CREATE TABLE GRADE_REPORT (
    Student_number INT,
    Section_identifier INT,
    Grade CHAR(1),
    PRIMARY KEY (Student_number, Section_identifier),
    FOREIGN KEY (Student_number) REFERENCES STUDENT(Student_number),
    FOREIGN KEY (Section_identifier) REFERENCES SECTION(Section_identifier)
);

CREATE TABLE PREREQUISITE (
    Course_number VARCHAR(10),
    Prerequisite_number VARCHAR(10),
    PRIMARY KEY (Course_number, Prerequisite_number),
    FOREIGN KEY (Course_number) REFERENCES COURSE(Course_number)
```

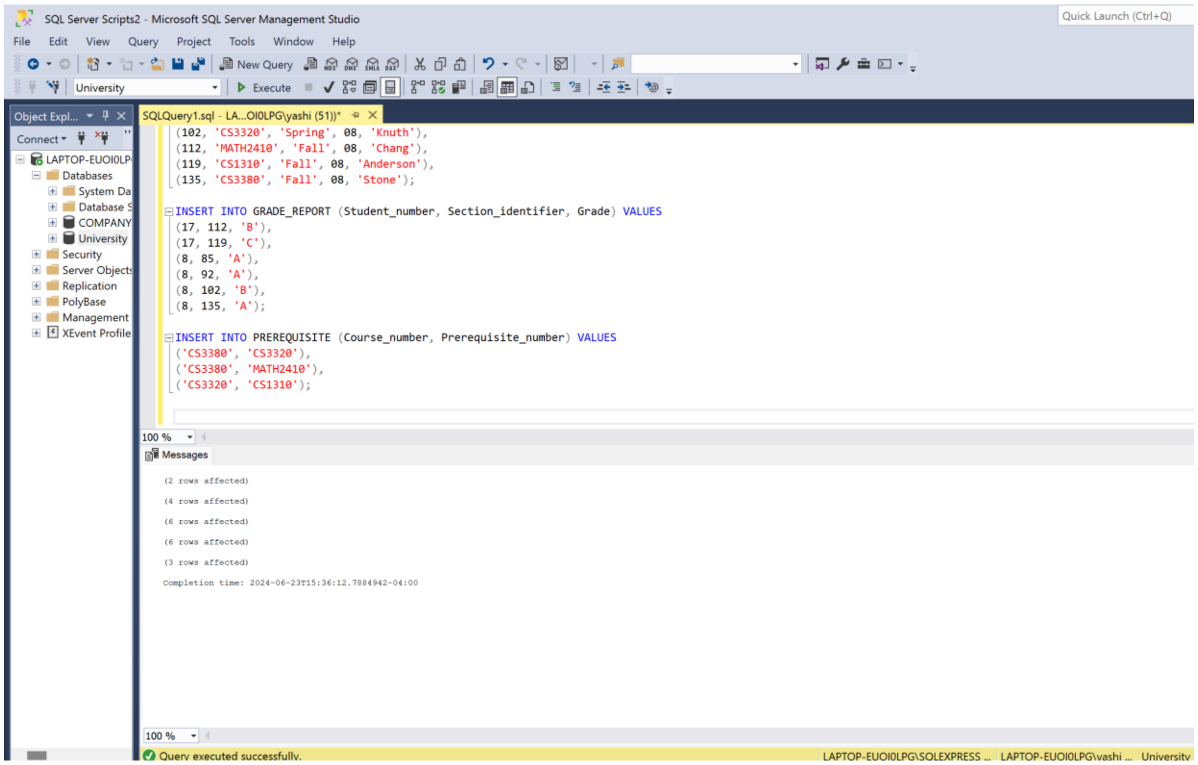
Next, the data is populated using ‘INSERT INTO’ commands into the tables as shown below:



```
    Prerequisite_number VARCHAR(10),
    PRIMARY KEY (Course_number, Prerequisite_number),
    FOREIGN KEY (Course_number) REFERENCES COURSE(Course_number),
    FOREIGN KEY (Prerequisite_number) REFERENCES COURSE(Course_number)
);

INSERT INTO STUDENT (Name, Student_number, Class, Major) VALUES
('Smith', 17, 1, 'CS'),
('Brown', 8, 2, 'CS');

INSERT INTO COURSE (Course_name, Course_number, Credit_hours, Department) VALUES
('Intro to Computer Science', 'CS1310', 4, 'CS'),
('Data Structures', 'CS3320', 4, 'CS'),
('Discrete Mathematics', 'MATH2410', 3, 'MATH'),
('Database', 'CS3380', 3, 'CS');

INSERT INTO SECTION (Section_identifier, Course_number, Semester, Year, Instructor) VALUES
(85, 'MATH2410', 'Fall', 07, 'King'),
(92, 'CS1310', 'Fall', 07, 'Anderson'),
(102, 'CS3320', 'Spring', 08, 'Knuth'),
(112, 'MATH2410', 'Fall', 08, 'Chang'),
(119, 'CS1310', 'Fall', 08, 'Anderson'),
(135, 'CS3380', 'Fall', 08, 'Stone');

INSERT INTO GRADE_REPORT (Student_number, Section_identifier, Grade) VALUES
(17, 112, 'B'),
(17, 119, 'C'),
(8, 85, 'A'),
(8, 92, 'A'),
(8, 102, 'B'),
(8, 135, 'A');

INSERT INTO PREREQUISITE (Course_number, Prerequisite_number) VALUES
('CS3380', 'CS3320'),
('CS3380', 'MATH2410'),
('CS3320', 'CS1310');
```

```
Connected. (1/1)
```

The following screenshot shows that the data has been populated successfully according to the database given.



```
(102, 'CS3320', 'Spring', 08, 'Knuth'),
(112, 'MATH2410', 'Fall', 08, 'Chang'),
(119, 'CS1310', 'Fall', 08, 'Anderson'),
(135, 'CS3380', 'Fall', 08, 'Stone');

INSERT INTO GRADE_REPORT (Student_number, Section_identifier, Grade) VALUES
(17, 112, 'B'),
(17, 119, 'C'),
(8, 85, 'A'),
(8, 92, 'A'),
(8, 102, 'B'),
(8, 135, 'A');

INSERT INTO PREREQUISITE (Course_number, Prerequisite_number) VALUES
('CS3380', 'CS3320'),
('CS3380', 'MATH2410'),
('CS3320', 'CS1310');
```

```
(2 rows affected)

(4 rows affected)

(6 rows affected)

(6 rows affected)

(3 rows affected)

Completion time: 2024-06-23T15:36:12.7884942-04:00
```

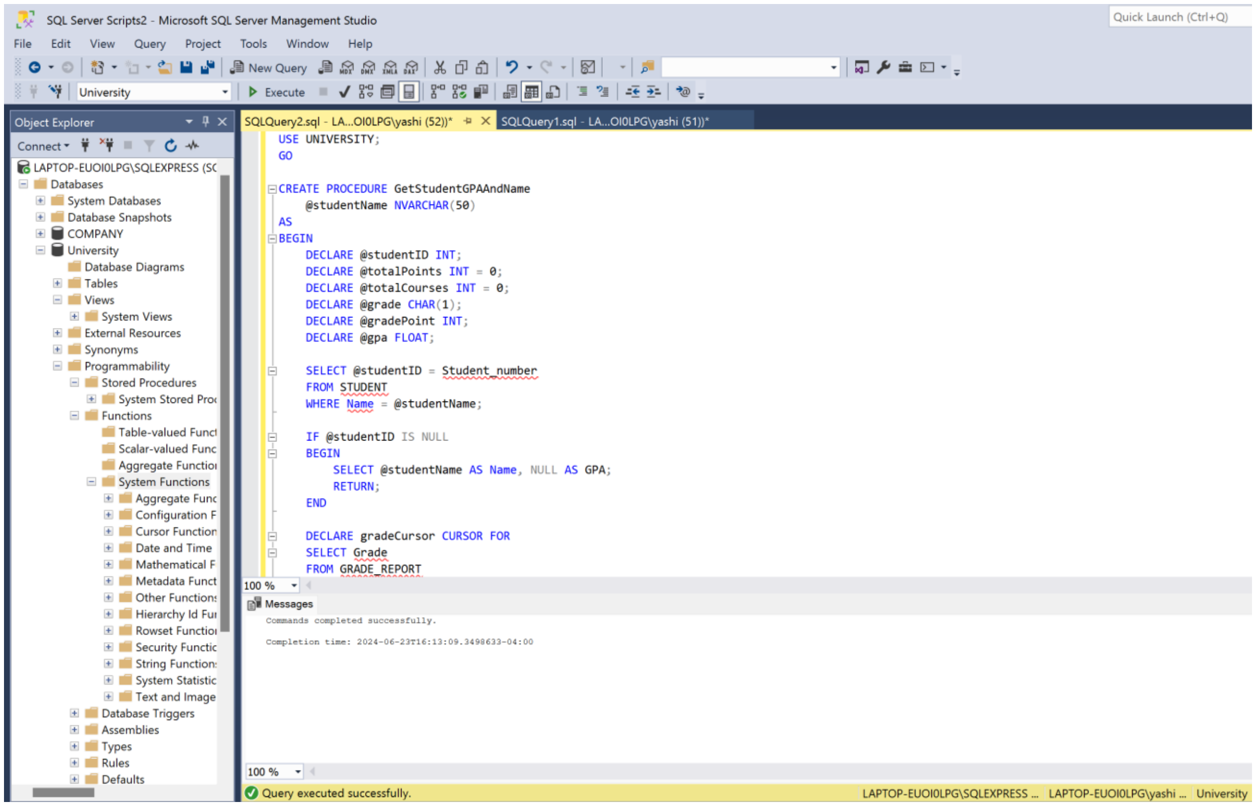
```
Query executed successfully.
```

2. **CREATING STORED PROCEDURE:**

I created a stored procedure called 'GetStudentGPAAndName' to calculate and return the GPA and name of the student given as input to the stored procedure.

The procedure, as shown below, gets the StudentID based on the specified student's name, checks if the student exists or not, uses a cursor to iterate through the student's grades and calculates the total points, closes the cursor and calculates the GPA. It returns the student's name along with the GPA as a table.

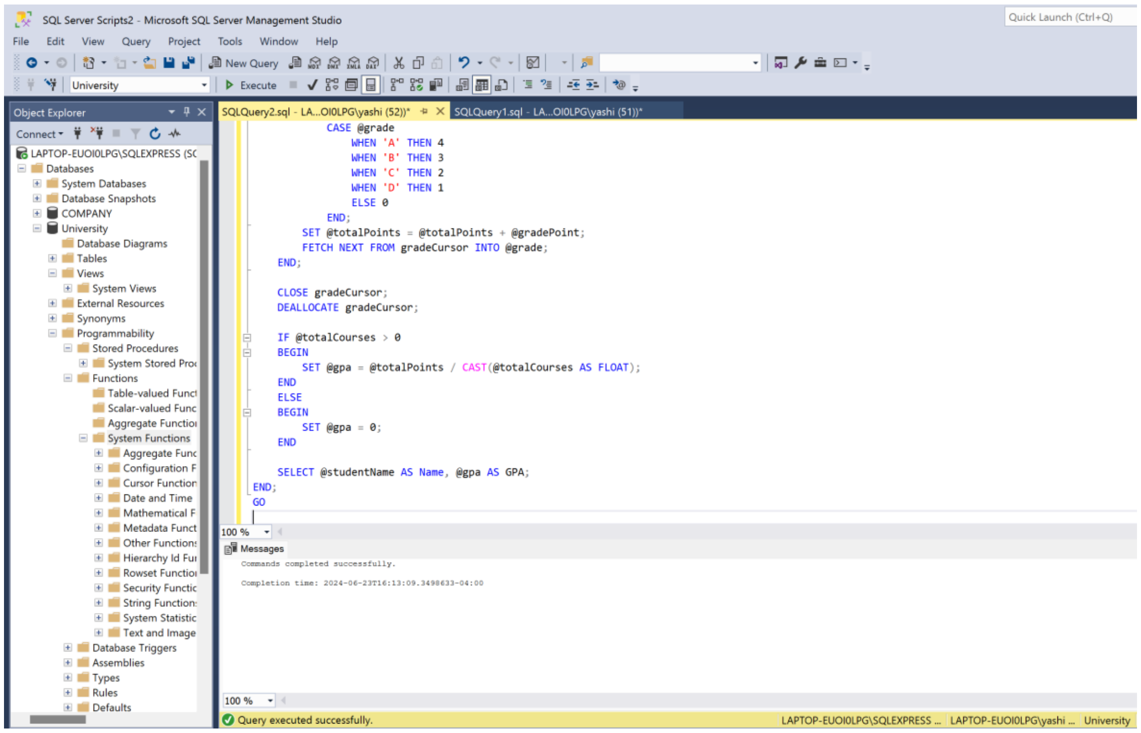
The screenshot below shows the execution of the stored procedure.



```sql
USE UNIVERSITY;
GO

CREATE PROCEDURE GetStudentGPAAndName
    @studentName NVARCHAR(50)
AS
BEGIN
    DECLARE @studentID INT;
    DECLARE @totalPoints INT = 0;
    DECLARE @totalCourses INT = 0;
    DECLARE @grade CHAR(1);
    DECLARE @gradePoint INT;
    DECLARE @gpa FLOAT;

    SELECT @studentID = Student_number
    FROM STUDENT
    WHERE Name = @studentName;

    IF @studentID IS NULL
    BEGIN
        SELECT @studentName AS Name, NULL AS GPA;
        RETURN;
    END

    DECLARE gradeCursor CURSOR FOR
    SELECT Grade
    FROM GRADE_REPORT
```

```
Commands completed successfully.

Completion time: 2024-06-23T16:13:09.3498633-04:00
```

Query executed successfully.

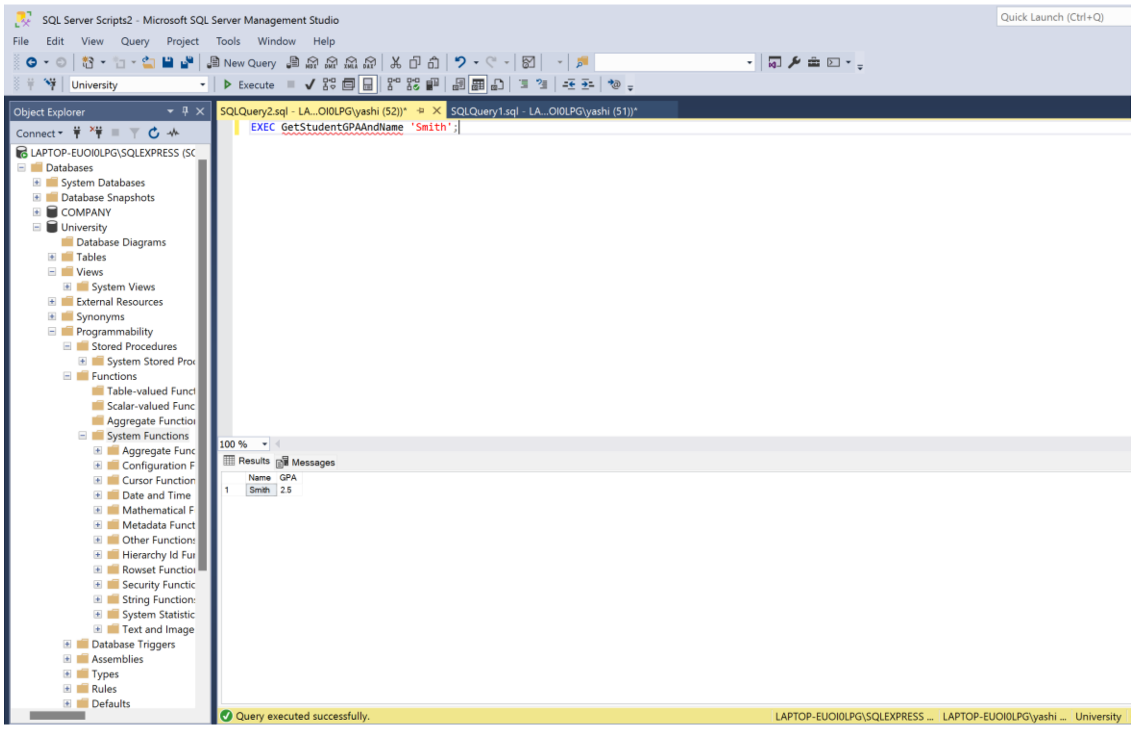


```
            CASE @grade
                WHEN 'A' THEN 4
                WHEN 'B' THEN 3
                WHEN 'C' THEN 2
                WHEN 'D' THEN 1
                ELSE 0
            END;
        SET @totalPoints = @totalPoints + @gradePoint;
        FETCH NEXT FROM gradeCursor INTO @grade;
    END;

    CLOSE gradeCursor;
    DEALLOCATE gradeCursor;

    IF @totalCourses > 0
    BEGIN
        SET @gpa = @totalPoints / CAST(@totalCourses AS FLOAT);
    END
    ELSE
    BEGIN
        SET @gpa = 0;
    END

    SELECT @studentName AS Name, @gpa AS GPA;
END;
GO
```

```
Commands completed successfully.

Completion time: 2024-06-23T16:13:09.3498633-04:00
```

Query executed successfully.

3. **<u>CALLING THE STORED PROCEDURE:</u>** To get the student's GPA along with name, I called the stored procedure by passing the student's name as a parameter as shown below. The command returned the GPA for the student named 'Smith' showing that the procedure successfully computes the GPA and returns it with name.



```
EXEC GetStudentGPAAndName 'Smith';
```

	Name	GPA
1	Smith	2.5

Query executed successfully.

Conclusion:

In this task, I successfully created a stored procedure in SQL Server to read a student's name and calculate their GPA based on the grades received in various courses, stored in the University database.

I began by creating the necessary tables using SQL ‘CREATE TABLE’ commands. Each table contains various attributes as specified in the University database attached to this lab's question.

Next, after setting the tables, I populated the data using ‘INSERT INTO’ commands.

The main task of the lab is to create a stored procedure, which I named ‘GetStudentGPAAndName’. This procedure utilizes the SQL Server’s procedural extensions to:

1. Take the StudentID based on the student's name. The procedure selects the Student_number from the STUDENT table based on the provided student name (@studentName). This ID is stored in @studentID.
2. Check if the student exists. If @studentID is ‘NULL’ it returns the input student name and NULL for the GPA.
3. Calculate the total grade points by iterating through student grades using cursor.
4. Count the number of courses taken by the student.
5. Compute the GPA by assigning numerical value based on letter grade and dividing the total grade points by the number of courses.
6. Return the GPA (@gpa) along with student name as a result.

I executed the stored procedure by specifying the student's name - ‘Smith’ to verify its functionality. The result was returned in a tabular format to represent the GPA corresponding to the student.

This lab demonstrates how effective SQL Server’s procedural extensions are in managing data. By using stored procedures, I was able to apply the logic for calculating GPA with a reusable database. This simplifies the queries and gives consistent results. Though the data in this database is not overly complicated, this approach could prove to be very useful in complex and larger volumes of data. Through this lab, I learned the practical applications of SQL Server’s procedural extensions and their capability to manipulate data within relational database system.

References:

1. https://learn.microsoft.com/en-us/sql/relational-databases/stored-procedures/create-a-stored-procedure?view=sql-server-ver16
2. https://youtu.be/xekOQXmSeJ4?si=wa6y0huzpn01rW_e
3. https://youtu.be/NrBJmtD0kEw?si=M3E-0titNu-ou00S
4. https://youtu.be/Sggdhot-MoM?si=AcLoL3PLqdexf869