**NAME: SIDDHARTH NAUTIYAL**

**COLLEGE: UPES**

**CSI ID:  CT-CSI23/SQ0108**

**TASK 1**

Write a Stored Procedure that takes table name as input parameter and finds out the name of all the columns in table.

**Constraint:** You cannot use sp\_columns statement

**Ans:** 1. First we create a database with name as sid

create database sid

2.Then Create the table as tblEmployee

CREATE TABLE tblEmployee (

ID INT PRIMARY KEY,

Name NVARCHAR(50),

Gender NVARCHAR(10),

Department\_ID INT

);

3. We Insert 10 random rows into the table

INSERT INTO tblEmployee (ID, Name, Gender, Department\_ID)

VALUES

(1, 'John', 'Male', 101),

(2, 'Alice', 'Female', 102),

(3, 'Bob', 'Male', 103),

(4, 'Emily', 'Female', 104),

(5, 'Michael', 'Male', 105),

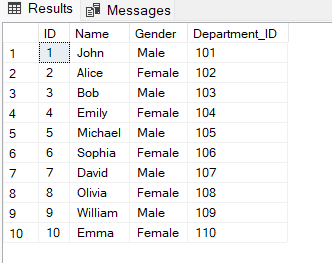
(6, 'Sophia', 'Female', 106),

(7, 'David', 'Male', 107),

(8, 'Olivia', 'Female', 108),

(9, 'William', 'Male', 109),

(10, 'Emma', 'Female', 110);



4. We create the Procedure named as GetColumnsInTable

CREATE PROCEDURE GetColumnsInTable

@TableName NVARCHAR(128)

AS

BEGIN

SELECT COLUMN\_NAME

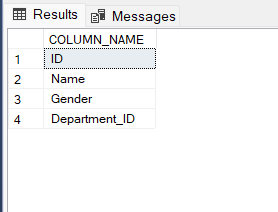
FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = @TableName

END;

5. And at last we execute the query by exec or execute

EXEC GetColumnsInTable @TableName = 'tblEmployee';

****

**TASK 2**

Write a Stored Procedure that inserts all the data using list of columns from Table A to Table B and truncates the data in Table A post insertion to Table B.

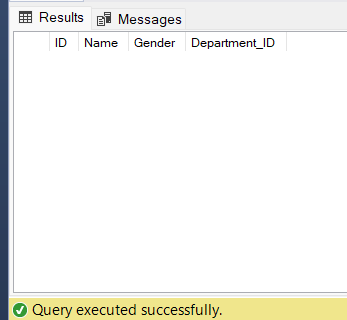
Constraint: The column list will be passed in a single parameter and the column values would be comma separated

**Ans:**

I am taking last table that we created in task 1 as table A which was tblEmployee

And the same database as sid.

So, now we have to make table B first with the same column as table A has.

1. First, we create a table named as table B

CREATE TABLE TableB (

ID INT,

Name NVARCHAR(50),

Gender NVARCHAR(10),

Department\_ID INT

);

1. Now we will create the procedure to add the values and truncate the table 1st

CREATE PROC InsertDataAndTruncate

@ColumnList NVARCHAR(MAX)

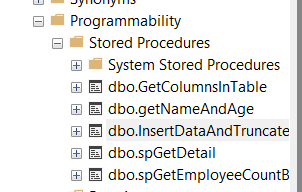
AS

BEGIN

INSERT INTO TableB (ID, Name, Gender, Department\_ID)

SELECT ID, Name, Gender, Department\_ID

FROM tblEmployee;



TRUNCATE TABLE tblEmployee;

END;

1. At last, we will execute the exec query

EXEC InsertDataAndTruncate @ColumnList = 'ID, Name, Gender, Department\_ID';

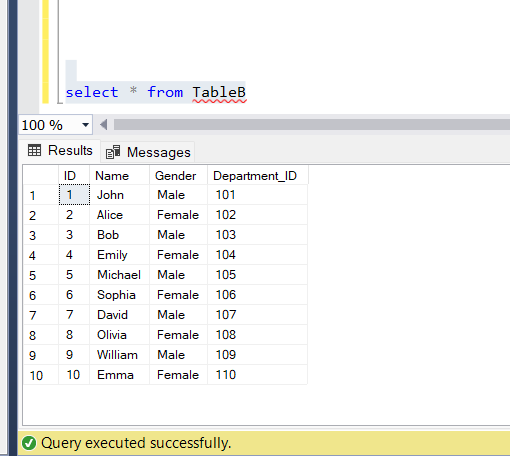
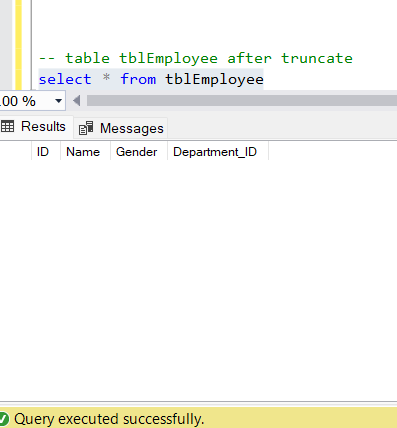


Table A or tblEmployee



**TASK 3**

**STUDENT ALLOTMENT**

According to the question provided to us there are 5 tables having different set of constraints and values.

In order to write the procedure for the given question we have to first make all of these tables and insert the particular data they hold.  
Since it is not given that we can make amendments on our own so I will work according the given question. I will take my old database i.e, ‘sid’ to work on.

For that lets start with creating tables:

1.StudentDetails and inserting values in it.

use sid

go

create table StudentDetails(

StudentID INT PRIMARY KEY,

StudentName NVARCHAR(50),

GPA DECIMAL(3,1),

Branch NVARCHAR(10),

Section VARCHAR(10)

);

insert into StudentDetails (StudentID,StudentName,GPA,Branch,Section) values

(159103036,'Mohit Agarwal', 8.9, 'CCE', 'A'),

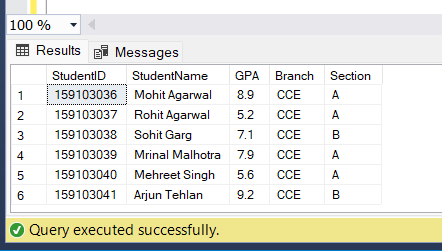
(159103037,'Rohit Agarwal', 5.2, 'CCE', 'A'),

(159103038,'Sohit Garg', 7.1, 'CCE', 'B'),

(159103039,'Mrinal Malhotra', 7.9, 'CCE', 'A'),

(159103040,'Mehreet Singh', 5.6,'CCE', 'A'),

(159103041,'Arjun Tehlan', 9.2, 'CCE', 'B');



2. SubjectDetails and inserting values in it

create table SubjectDetails(

SubjectID NVARCHAR(60) PRIMARY KEY,

SubjectName NVARCHAR(50),

MaxSeats INT,

RemainingSeats INT

);

insert into SubjectDetails (SubjectID, SubjectName, MaxSeats, RemainingSeats) values

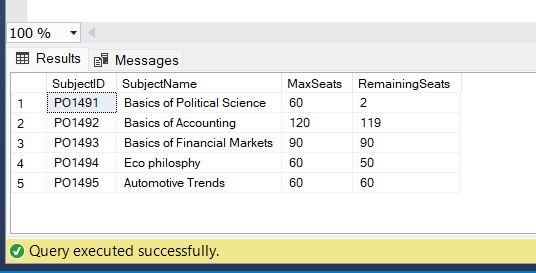
('PO1491', 'Basics of Political Science', 60, 2),

('PO1492', 'Basics of Accounting', 120, 119),

('PO1493', 'Basics of Financial Markets', 90, 90),

('PO1494', 'Eco philosphy', 60, 50),

('PO1495', 'Automotive Trends', 60, 60);



3.StudentPreference and inserting values in it.

create table StudentPreference(

StudentId INT,

SubjectId nvarchar(60) ,

Preference int,

foreign key(StudentId)

references StudentDetails(StudentID),

foreign key(SubjectId)

references SubjectDetails(SubjectID)

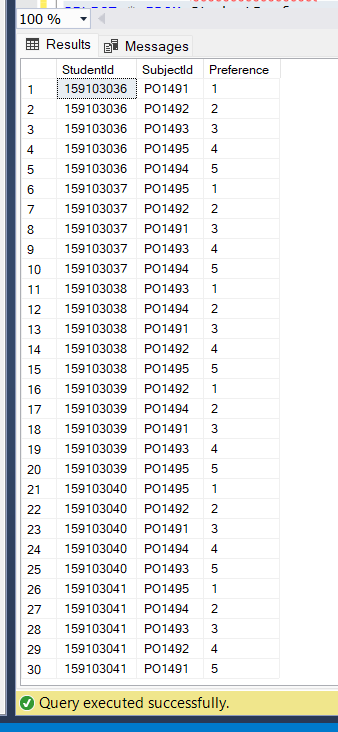
);

INSERT INTO StudentPreference (StudentId, SubjectId, Preference)

VALUES

INSERT INTO StudentPreference (StudentId, SubjectId, Preference)

VALUES

(159103036, 'PO1491', 1),

(159103036, 'PO1492', 2),

(159103036, 'PO1493', 3),

(159103036, 'PO1494', 4),

(159103036, 'PO1495', 5),

(159103037, 'PO1495', 1),

(159103037, 'PO1492', 2),

(159103037, 'PO1491', 3),

(159103037, 'PO1493', 4),

(159103037, 'PO1494', 5),

(159103038, 'PO1493', 1),

(159103038, 'PO1494', 2),

(159103038, 'PO1491', 3),

(159103038, 'PO1492', 4),

(159103038, 'PO1495', 5),

(159103039, 'PO1492', 1),

(159103039, 'PO1494', 2),

(159103039, 'PO1491', 3),

(159103039, 'PO1493', 4),

(159103039, 'PO1495', 5),

(159103040, 'PO1495', 1),

(159103040, 'PO1492', 2),

(159103040, 'PO1491', 3),

(159103040, 'PO1494', 4),

(159103040, 'PO1493', 5),

(159103041, 'PO1495', 1),

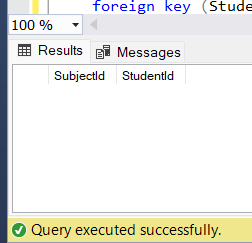
(159103041, 'PO1494', 2),

(159103041, 'PO1493', 3),

(159103041, 'PO1492', 4),

(159103041, 'PO1491', 5);

Since, there are a total of 6 students and have 5 preferences so there a total of 30 entries in the preference table.

4. Allotment table

create table Allotment(

SubjectId nvarchar(60),

StudentId int unique,

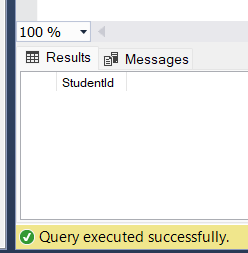
foreign key (SubjectId)

references SubjectDetails(SubjectID),

foreign key (StudentId)

references StudentDetails(StudentID)

);



5. UnallotedStudents

create table UnallotedStudents(

StudentId int unique,

foreign key (StudentId)

references StudentDetails(StudentID)

);

**LOGIC:**

Now, comes the major and most important part that is to make a stored procedure of the given task.

Let’s just understand the logic behind the query through steps:

* First, we will sort the table(#StudentPreferences) on the basis of GPA. In my case I inserted values in sorted form.
* second, since preferences are already in ordered form (1,2,3,4,5) so we don't have to make it in ordered way and directly check the preference and remaining seats.
* if there is seat available then we will allot it with studentID and subjectID in allotment table and deduct a single digit from the remaining seat of StudentDetail table.
* Else we will move to the second preference and third and till the the last preference that is 5th
* If all the remaining seats are full then we update the unalloted table with that studentID and repeat again the process till every studentID has been checked.

Let’s just jump into stored procedure Part:

CREATE PROCEDURE AllocateSubjects

AS

BEGIN

CREATE TABLE #StudentPreferences (

StudentId INT,

SubjectId NVARCHAR(60),

Preference INT,

GPA DECIMAL(3, 1)

);

-- Inserting data into the temporary table

INSERT INTO #StudentPreferences (StudentId, SubjectId, Preference, GPA)

VALUES

(159103041, 'PO1495', 1, 9.2),

(159103041, 'PO1494', 2, 9.2),

(159103041, 'PO1493', 3, 9.2),

(159103041, 'PO1492', 4, 9.2),

(159103041, 'PO1491', 5, 9.2),

(159103036, 'PO1491', 1, 8.9),

(159103036, 'PO1492', 2, 8.9),

(159103036, 'PO1493', 3, 8.9),

(159103036, 'PO1494', 4, 8.9),

(159103036, 'PO1495', 5, 8.9),

(159103039, 'PO1492', 1, 7.9),

(159103039, 'PO1494', 2, 7.9),

(159103039, 'PO1491', 3, 7.9),

(159103039, 'PO1493', 4, 7.9),

(159103039, 'PO1495', 5, 7.9),

(159103038, 'PO1493', 1, 7.1),

(159103038, 'PO1494', 2, 7.1),

(159103038, 'PO1491', 3, 7.1),

(159103038, 'PO1492', 4, 7.1),

(159103038, 'PO1495', 5, 7.1),

(159103040, 'PO1495', 1, 5.6),

(159103040, 'PO1492', 2, 5.6),

(159103040, 'PO1491', 3, 5.6),

(159103040, 'PO1494', 4, 5.6),

(159103040, 'PO1493', 5, 5.6),

(159103037, 'PO1495', 1, 5.2),

(159103037, 'PO1492', 2, 5.2),

(159103037, 'PO1491', 3, 5.2),

(159103037, 'PO1493', 4, 5.2),

(159103037, 'PO1494', 5, 5.2);

DECLARE @NextPreference INT = 1;

DECLARE @SubjectId NVARCHAR(60);

DECLARE @RemainingSeats INT;

WHILE @NextPreference <= 5 -- since there are 5 preferences for each student

BEGIN

SELECT TOP 1 @SubjectId = SD.SubjectId, @RemainingSeats = SD.RemainingSeats

FROM SubjectDetails SD

JOIN #StudentPreferences SP ON SD.SubjectId = SP.SubjectId

WHERE SP.Preference = @NextPreference

AND NOT EXISTS (SELECT 1 FROM Allotment A WHERE A.StudentId = SP.StudentId)

ORDER BY

CASE

WHEN SD.RemainingSeats = 0 THEN @NextPreference + 1

ELSE @NextPreference

END,

SP.GPA DESC;

-- this part of code is the main part where most of things are done

INSERT INTO Allotment (SubjectId, StudentId)

SELECT TOP 1 SP.SubjectId, SP.StudentId

FROM #StudentPreferences SP

JOIN SubjectDetails SD ON SP.SubjectId = SD.SubjectId

WHERE SP.Preference = @NextPreference AND SP.SubjectId = @SubjectId

AND SD.RemainingSeats > 0

AND NOT EXISTS (SELECT 1 FROM Allotment A WHERE A.StudentId = SP.StudentId)

ORDER BY SP.GPA DESC;

-- allotment to the studentID which have highest preference and highest GPA

-- Checking if any student has been allocated in this preference

IF @@ROWCOUNT =0

BEGIN

SET @NextPreference = @NextPreference + 1;

END

ELSE

BEGIN

UPDATE SubjectDetails SET RemainingSeats = RemainingSeats - 1

WHERE SubjectId = @SubjectId AND RemainingSeats > 0;

SET @NextPreference = 1;

END;

END;

-- Inserting unalloted students into the UnallotedStudents table

INSERT INTO UnallotedStudents (StudentId)

SELECT DISTINCT StudentId FROM #StudentPreferences WHERE StudentId NOT IN (SELECT StudentId FROM Allotment);

-- Droping the temporary table because we don’t need it anymore

DROP TABLE #StudentPreferences;

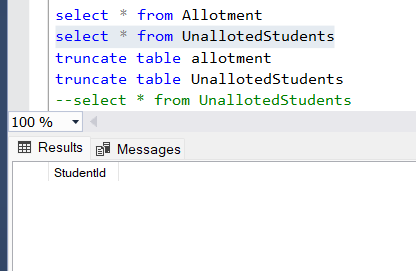
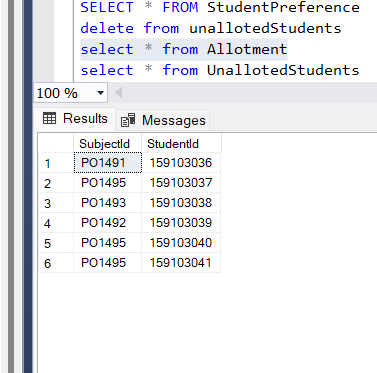
PRINT 'Subject Allocation Process Completed.';

END;

-- with this procedure is complete and for executing we simply can execute by using exec

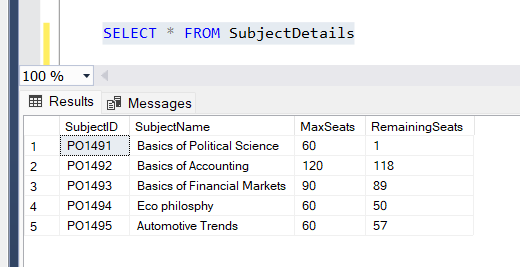
EXEC AllocateSubjects

Now, let’s see the output part of the procedure



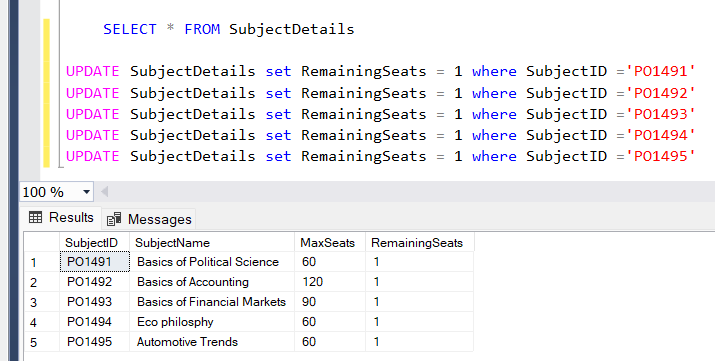
since, there were many remaining seats, because of that every student was allotted in the allotment table. And we little cautiously we find that student is in descending form of GPA.

Now let’s have a look in the subjectDetail as well



We see that 6 seats were deducted from equally

It means that everything is running fine, but what if there is only 1 seat for every subject, we are doing this because we want to know whether the Unalloted table is working fine or not. Let’s have a look at it as well

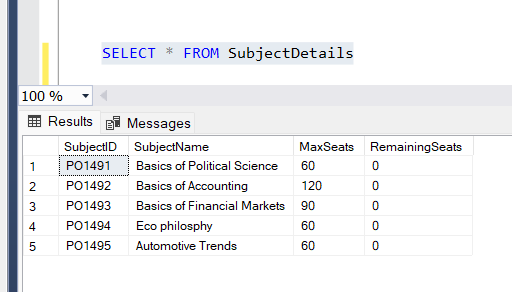
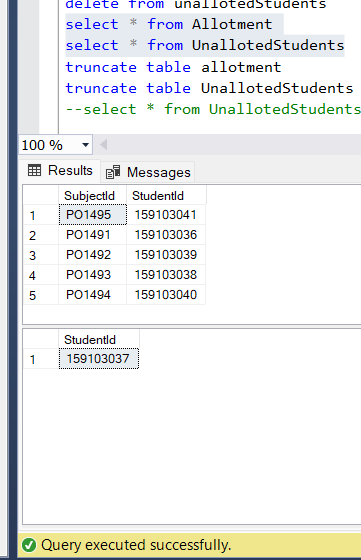


So first we update the values of remaining seats

Then before execute we truncate the the allotment table and unallotment table so that there should not remain any ambiguity.

Then after executing we get the desired result

From this output we can see that every table is updating finely



If look closer we find that studentID ending with 37 has the lowest GPA. And because of it was not provided any subject from the table.

In this task only I learned various new thing i.e., making temporary table using ‘#’, use of while loop, use of case statement, use of @@rowcount function as well.

**SUBJECT CHANGE REQUEST**

In this task we have to allot a different subject to a student who wants to change its earlier subject. For this we were provided 2 different tables in which we have show case the data of changed subject and allotted subject. Before making a project we first have to define the table.  
let’s make table first

**First, we will make SubjectAllotment table and insert values in it:**

create table SubjectAllotments

( StudentID int,

SubjectID nvarchar(60),

Is\_Valid bit,

);

insert into SubjectAllotments values

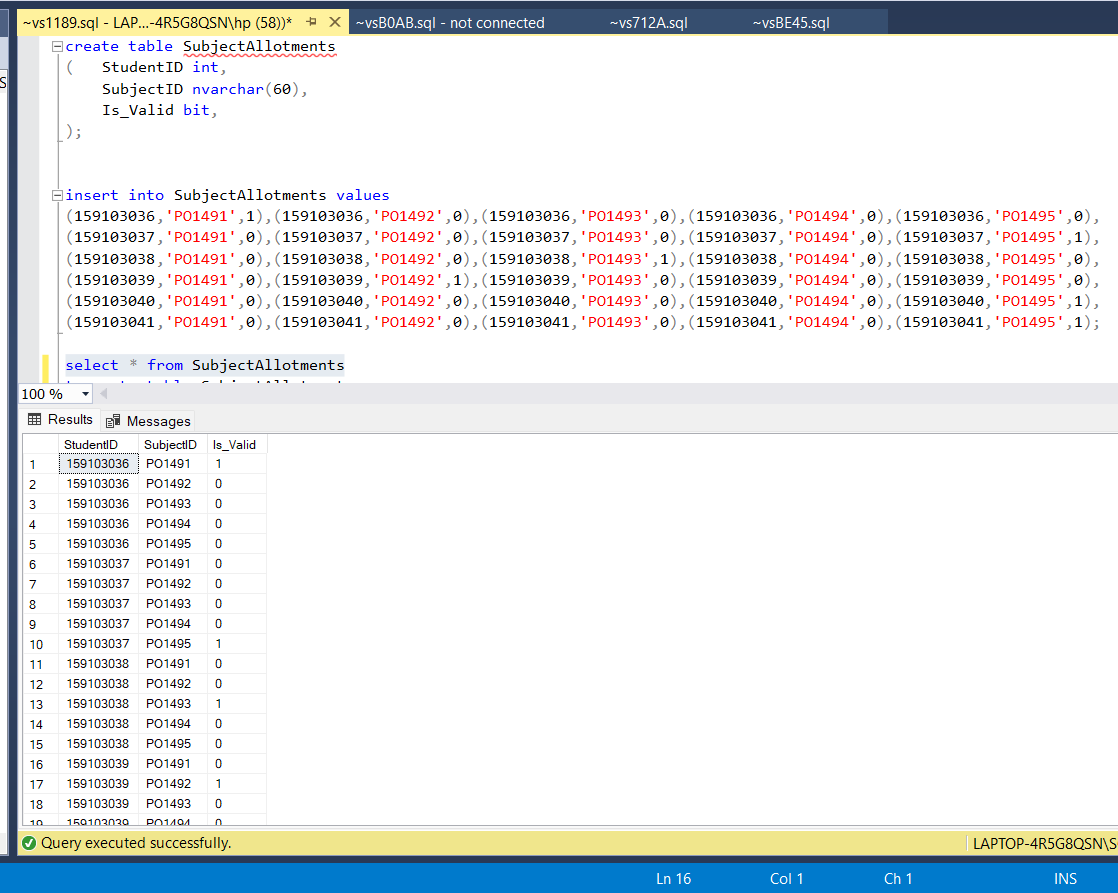
(159103036,'PO1491',1), (159103036,'PO1492',0), (159103036,'PO1493',0), (159103036,'PO1494',0), (159103036,'PO1495',0),

(159103037,'PO1491',0), (159103037,'PO1492',0), (159103037,'PO1493',0), (159103037,'PO1494',0), (159103037,'PO1495',1),

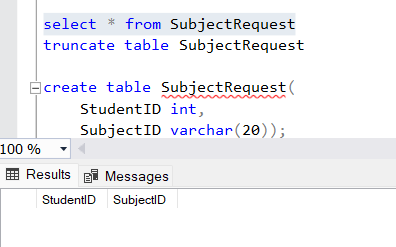
(159103038,'PO1491',0), (159103038,'PO1492',0), (159103038,'PO1493',1), (159103038,'PO1494',0), (159103038,'PO1495',0),

(159103039,'PO1491',0), (159103039,'PO1492',1), (159103039,'PO1493',0), (159103039,'PO1494',0), (159103039,'PO1495',0),

(159103040,'PO1491',0), (159103040,'PO1492',0), (159103040,'PO1493',0), (159103040,'PO1494',0), (159103040,'PO1495',1),

(159103041,'PO1491',0), (159103041,'PO1492',0), (159103041,'PO1493',0), (159103041,'PO1494',0), (159103041,'PO1495',1);

These are the subject allotment which we did in the 3rd task in the allotment table and Is\_valid is there preferences of the students which they choose.

**Next, we create table SubjectRequest:**

create table SubjectRequest(

StudentID int,

SubjectID varchar(20));

Now, comes the Important part that is to make procedure for the mentioned task. Before that let us understand the logic behind this and know how this procedure actually works.

**LOGIC:**

* First, we will check if the student exists in SubjectAllotments.
* Then we will check if the requested subject is different from the current valid subject.
* If yes, then we will update the SubjectRequest table with the StudentID and SubjectID and also update the is\_valid column of SubjectAllotments.
* If no, then we will do nothing
* If the studentID is different from the given table then we simply insert the requested subject as a valid record in the SubjectAllotments Table.

Let’s jump on the stored procedure now,

ALTER PROCEDURE ProcessSubjectChangeRequest

@StudentID int,

@SubjectID VARCHAR(50)

AS

BEGIN

IF EXISTS (SELECT 1 FROM SubjectAllotments WHERE StudentID = @StudentID)

BEGIN

IF EXISTS (SELECT 1 FROM SubjectAllotments WHERE StudentID = @StudentID AND SubjectID! = @SubjectID AND Is\_Valid = 1)

BEGIN

DECLARE @PreviousSubjectID VARCHAR(50);

SELECT @PreviousSubjectID = SubjectID

FROM SubjectAllotments

WHERE StudentID = @StudentID AND Is\_Valid = 1;

UPDATE SubjectAllotments

SET Is\_Valid = 0

WHERE StudentID = @StudentID AND Is\_Valid = 1;

INSERT INTO SubjectRequest (StudentID, SubjectID)

VALUES (@StudentID, @PreviousSubjectID);

UPDATE SubjectAllotments

SET Is\_Valid = 1

WHERE StudentID = @StudentID AND SubjectID = @SubjectID;

END;

END

ELSE

BEGIN

INSERT INTO SubjectAllotments (StudentID, SubjectID, Is\_Valid)

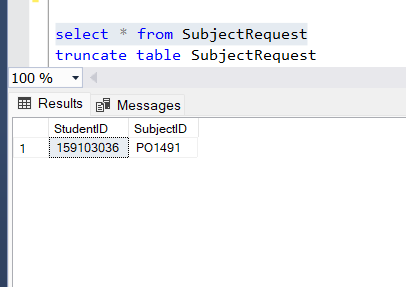
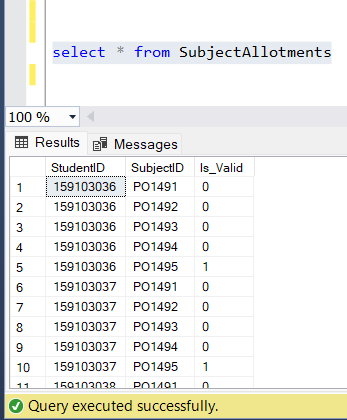
VALUES (@StudentID, @SubjectID, 1);

END;

END;

We will execute the procedure by passing the parameter in it:

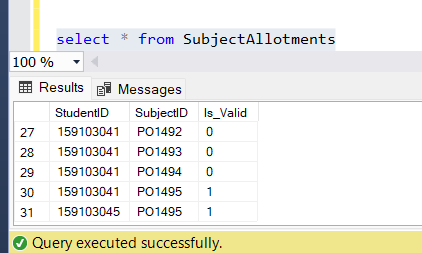
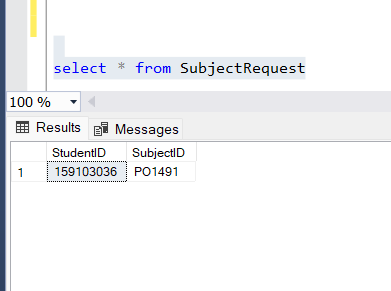
EXEC ProcessSubjectChangeRequest @StudentID = 159103036, @SubjectID = 'PO1495';

Let’s see the what the output is showing to us:

We see that both the tables are working fine, now let us try with few more entries i.e., changing the studentID to a different value. let say (159103045)

EXEC ProcessSubjectChangeRequest @StudentID = 159103045, @SubjectID = 'PO1495';

After executing we see that a new value was added in the table SubjectAllotment and no value was added in the SubjectRequest because there was no replacement in it.

We have tried both the method that the task was asking and my test cases run were fine.

And hence this marks the end of the project.

**Thank you**