

## MSBA230: Final Project

Our project uses the Research database of the Corporate Affiliate program. The first question that we wanted to answer was: “How can students find available projects for them to join?” For this question, we wanted to put limits on what projects students were able to join. If they were below a 2.75 GPA, then they were not allowed to join any projects. Or if there were already five students in a project, then that meant there was no availability to join. Lastly, students were allowed to input their major to find projects related to what they studied.

The stored procedure that was made to answer our first question can be used by students who do not have any projects yet but are trying to get one. There are two user inputs, the student ID, and the student's major. Once the user provides these two variables, the program will tell whether the given student's major has any ongoing project.

First, we named the stored procedure `spProjectFinder` since the program was made so students can find projects. There are two variables defined here; one of them is `@studentID`, and the other one is `@major`. The user can input these two variables by themselves.

```
alter proc spProjectFinder
    @studentID int,
    @major nvarchar(50)
as
begin
```

Secondly, we declared two more variables, `@studentGPA`, which stores the given student's GPA, and `@projectCount`, which stores the number of projects available for a given major.

```
begin
    declare @studentGPA decimal(3,2)
    declare @projectCount int
```

This block connects the 'StudentID' with the user input and then uses @studentGPA, which will get the given student's GPA. Then, the program will check whether the student has a higher or lower GPA than 2.75, which is the minimum criterion for being eligible to participate in projects. If the student has a lower GPA, the program raises an error message.

```
--Get student's GPA
select @studentGPA=StudentGPA
from Research.Student
where StudentID=@studentID
--Check if student is eligible based on GPA
if @studentGPA < 2.75
begin
    throw 50001, 'The student is not eligible to participate in any projects due to a low GPA.', 1
    return
end
```

This block checks how many ongoing projects are available for a given major. It checks the beginning date and the end date and gives results according to the user's current date. If there is not a single project for that given major, the program will raise an error message, letting the user know, that there are not any projects for the major currently.

```
--Check if the given major has projects
select @projectCount=count(p.ProjectID)
from Research.Project p join Research.Company c on p.CompanyID=c.CompanyID join Research.StudentProject sp on p.ProjectID=sp.ProjectID
where p.ProjectStartDate <= getdate() and p.ProjectType=@major and sp.EndDate > getdate()--Check if the project is active

if @projectCount=0
begin
    throw 50002, 'The given major currently has no projects.', 2
    return
end
```

Lastly, this block checks if a project has any spots for a new team member. If there are five people in a project, that project will not appear since there are no openings.

In the last two steps, the program associates the major with the 'ProjectType', and sees if there are any matches since the 'ProjectType' is the short description for students where they can check which projects are available for their majors.

```
--Get available projects with available spots
select p.ProjectID, p.ProjectType, count(sp.StudentID) as 'Number Of Members'
from Research.Project p left join Research.StudentProject sp on p.ProjectID=sp.ProjectID
where p.ProjectType=@major
group by p.ProjectID, p.ProjectType
having count(sp.StudentID)<5 --Limit to projects with available spots
order by p.ProjectID

end
```

If the user puts in:

```
exec spProjectFinder 12, 'Marketing'
```

	ProjectID	ProjectType	Number Of Members
1	21	Marketing	1

If the user puts in:

```
exec spProjectFinder 11, 'History'
```

Messages	
Msg 50001, Level 16, State 1, Procedure spProjectFinder, Line 15 [Batch Start Line 38]	The student is not eligible to participate in any projects due to a low GPA.
Completion time: 2023-12-08T11:34:56.5856254-08:00	

If the user puts in:

```
exec spProjectFinder 2, 'Biology'
```

Messages	
Msg 50002, Level 16, State 2, Procedure spProjectFinder, Line 25 [Batch Start Line 38]	The given major currently has no projects.
Completion time: 2023-12-08T11:35:46.3010922-08:00	

The second question our group asked was: “How to find out the status of a given project?”. Our goal is essentially to let the user input a project title, and the output will inform on the current status of the project; whether it has been finished, is active, is upcoming, or doesn't exist.

The procedure is named spProjectStatus, since it will be returning the different statuses of projects. The first variable is @ProjectTitle, which will be used as the input for the user to quickly look up the various projects. The second variable declared is @CurrentDate which when using the code, takes the current date for checking status.

```

ALTER PROCEDURE spProjectStatus
    @ProjectTitle NVARCHAR(100)
AS
BEGIN
    DECLARE @CurrentDate DATE = GETDATE();

```

When inserting a project title, if it exists, we wanted the output to give the Project ID, Project Type, and Project Title along with the defined status.

Then, using CASE statements allows us to evaluate our expression against multiple values and give the corresponding result. This allows us to define the different types of status of the project based on the student's end date. If the end date is less than the current date, then the project is 'Finished.' If the join date is less than or equal to the current date and the end date is greater than or equal to the current date, then the status is 'Active.' Lastly, if the join date exceeds the current date, then the status is 'Upcoming'.

We had to do a join between the projects table and the student project table for the query to properly perform.

Lastly, using the if-else statements, we have the code print an error message if the project title doesn't exist. This is because when looking up project status, someone may mistype the title or have the wrong title.

```

IF EXISTS (SELECT 1 FROM Research.Project WHERE ProjectTitle = @ProjectTitle)
BEGIN
    SELECT
        P.ProjectID,
        P.ProjectType,
        P.ProjectTitle,
        CASE
            WHEN SP.EndDate < @CurrentDate THEN 'Finished'
            WHEN SP.JoinDate <= @CurrentDate AND SP.EndDate >= @CurrentDate THEN 'Active'
            WHEN SP.JoinDate > @CurrentDate THEN 'Upcoming'
        END AS ProjectStatus
    FROM
        Research.Project AS P
    INNER JOIN
        Research.StudentProject AS SP ON P.ProjectID = SP.ProjectID
    WHERE
        P.ProjectTitle = @ProjectTitle;
END
ELSE
BEGIN
    PRINT 'Error: Project title does not exist';
END
END;

```

When executing spProjectStatus with a project title that doesn't exist, the following is the result:

```
EXEC spProjectStatus 'Uncommon Project';
```

9 %

Messages

Error: Project title does not exist

Completion time: 2023-12-12T11:49:47.4815687-08:00

When executing `spProjectStatus` with a project title that exists, the output gives the project type, title, project ID, and status.

```
EXEC spProjectStatus 'Increase Engagement';
```

Results Messages

ProjectID	ProjectType	ProjectTitle	ProjectStatus
1	Social Media	Increase Engagement	Finished

The last question we wanted to ask was: “Based on knowing a student or advisor's name, how do we find the projects they are assigned to?” Our goal with asking this question is by having one’s name then we will be able to find the various projects that they are assigned. The output we wanted to find for each person was their assigned project ID, title, type, and assignment status.

To begin, we named the stored procedure ‘`spProjectAssignment`.’ Our first variable declared is `@PersonName`, which represents a student or advisor's full name.

```
ALTER PROCEDURE spProjectAssignment
    @PersonName NVARCHAR(100)
```

The procedure begins by using the `CONCAT` function to combine the student's first and last name to represent `@PersonName`. If the name isn’t on the student project list, a message stating that they are not assigned to a project pops up.

```
BEGIN
    IF EXISTS (SELECT 1 FROM Research.Student WHERE CONCAT(StudentName, ' ', StudentLName) = @PersonName)
    BEGIN
        IF NOT EXISTS (SELECT 1 FROM Research.Student AS S
            LEFT JOIN Research.StudentProject AS SP ON S.StudentID = SP.StudentID
            WHERE CONCAT(S.StudentName, ' ', S.StudentLName) = @PersonName AND SP.ProjectID IS NOT NULL)
        BEGIN
            PRINT 'Not assigned to a project';
        END
    END
```

But if the student's name does appear, then the output will return the ‘ProjectID,’ ‘ProjectType,’ ‘ProjectTitle,’ and their ‘AssignmentStatus.’ To find this information, we must do a three-way join with the student table, student project table, and project table.

```

ELSE
BEGIN
    SELECT
        SP.ProjectID,
        P.ProjectType,
        P.ProjectTitle,
        'Assigned' AS AssignmentStatus
    FROM Research.Student AS S
    LEFT JOIN Research.StudentProject AS SP ON S.StudentID = SP.StudentID
    LEFT JOIN Research.Project AS P ON SP.ProjectID = P.ProjectID
    WHERE CONCAT(S.StudentName, ' ', S.StudentLName) = @PersonName;
END

```

We repeat our early step of using CONCAT to define @PersonName for the advisors. If the advisor's name is absent, then a message stating that is outputted.

```

ELSE IF EXISTS (SELECT 1 FROM Research.Advisor WHERE CONCAT(FName, ' ', LName) = @PersonName)
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Research.Advisor AS A
        LEFT JOIN Research.Comment AS C ON A.AdvisorID = C.AdvisorID
        LEFT JOIN Research.Project AS P ON C.ProjectID = P.ProjectID
        LEFT JOIN Research.StudentProject AS SP ON P.ProjectID = SP.ProjectID
        WHERE CONCAT(A.FName, ' ', A.LName) = @PersonName AND P.ProjectID IS NOT NULL)
    BEGIN
        PRINT 'Not assigned to a project';
    END
END

```

For advisor's names that are assigned to a project, we used CASE statements again to set our criteria. To find all the information on the advisor's assignment to a project, we had to do a four-way join with the advisor table, comment table, project table, and student project table. If there were no values for 'ProjectTitle', 'ProjectID,' and 'ProjectType,' then we conclude that the advisor is not assigned to a project.



```

ELSE
BEGIN
    SELECT
        CASE
            WHEN P.ProjectID IS NULL THEN NULL
            ELSE SP.ProjectID
        END AS ProjectID,
        CASE
            WHEN P.ProjectType IS NULL THEN 'Not assigned to a project'
            ELSE P.ProjectType
        END AS ProjectType,
        CASE
            WHEN P.ProjectTitle IS NULL THEN 'Not assigned to a project'
            ELSE P.ProjectTitle
        END AS ProjectTitle,
        CASE
            WHEN P.ProjectID IS NOT NULL THEN 'Assigned'
            ELSE 'Not assigned to a project'
        END AS AssignmentStatus
    FROM Research.Advisor AS A
    LEFT JOIN Research.Comment AS C ON A.AdvisorID = C.AdvisorID
    LEFT JOIN Research.Project AS P ON C.ProjectID = P.ProjectID
    LEFT JOIN Research.StudentProject AS SP ON P.ProjectID = SP.ProjectID
    WHERE CONCAT(A.FName, ' ', A.LName) = @PersonName;
END

```

The last portion is that if the advisor's name is not in the database, the output will print that the person is not found.

```

ELSE
BEGIN
    PRINT 'Person not found';
END

```

When executing spProjectAssignment and a person doesn't exist in the database:

The screenshot shows a SQL query window with the command `EXEC spProjectAssignment 'Anna Cable';`. Below the query window, the Messages pane displays the output: `Person not found`. The Completion time is shown as `2023-12-12T12:30:13.8046060-08:00`.

When executing spProjectAssignment and a person is yet to be assigned to a project:

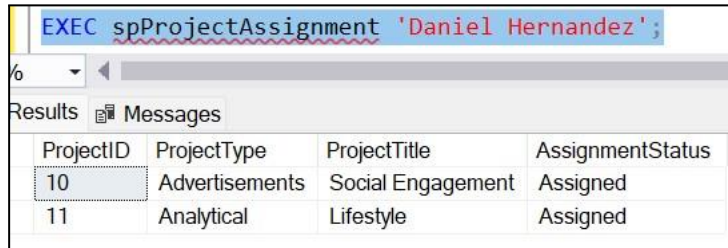
The screenshot shows a SQL query window with the command `EXEC spProjectAssignment 'Sandra Savage';`. Below the query window, the Messages pane displays the output: `Not assigned to a project`. The Completion time is shown as `2023-12-12T12:29:49.5230735-08:00`.

Oliver Fodor

Hailey Williams

Mansha Gupta

When executing spProjectAssignment and the person is assigned:



The screenshot shows a SQL Server Enterprise Manager interface. At the top, a query window displays the command `EXEC spProjectAssignment 'Daniel Hernandez';`. Below the query window, the 'Results' tab is active, showing a table with four columns: ProjectID, ProjectType, ProjectTitle, and AssignmentStatus. The table contains two rows of data. The first row has ProjectID 10, ProjectType Advertisements, ProjectTitle Social Engagement, and AssignmentStatus Assigned. The second row has ProjectID 11, ProjectType Analytical, ProjectTitle Lifestyle, and AssignmentStatus Assigned.

ProjectID	ProjectType	ProjectTitle	AssignmentStatus
10	Advertisements	Social Engagement	Assigned
11	Analytical	Lifestyle	Assigned