Project 3

# Project Info

## Due Date

Friday, November 20th, 11:59PM

## Submission

Submit a single “.sql” file to the Project 3 drop box.

You may submit multiple times to the dropbox. Only the most recent “.sql” file submission will be graded.

**This is an individual assignment. Significant overlap between submissions from two or more students may be flagged for plagiarism.**

## Grading

The project is graded out of 20 total marks & will be **worth 10%** of your final grade. Each requirement has an associated grade in its description.

Each requirement will be graded independently, though most requirements are at least partially dependent on the successful completion of previous steps.

## Initialization

**This project must run against the database created by "Project3\_InitialDB.sql" in the Project 3 folder.**

## Additional Info

Please submit all work in a single file and use comments to clarify where your work for a requirement begins and ends.

For example:

/\* REQUIREMENT 1 \*/

SELECT \*

FROM dbo.JustAnExample;

/\* REQUIREMENT 2 \*/

SELECT \*

FROM dbo.AlsoAnExample;

**You are encouraged to test your code before submitting (but please do not submit any of your extra tests – just the ones requested below) . Your script must execute without error against the database created by "Project3\_InitialDB.sql".**

# Requirements

## Requirement 1 – Basic Stored Procedure ( 1 Mark )

Create a stored procedure to insert into dbo.Departments.

The procedure should accept the appropriate parameters (only those required to create the record).

## Requirement 2 – Basic Procedure Execution ( 1 Mark )

Write a script that will execute the procedure created in requirement 1. Create the following three departments:

|  |  |
| --- | --- |
| DepartmentName | DepartmentDesc |
| SQA | Software Testing and Quality Assurance |
| Development | Systems Design and Development |
| Deployment | Deployment and Production Support |
| TechSupport | Online Technical Support |

## Requirement 3 – Scalar Function ( 2 Mark )

Create a function to get an Department ID by name (not Desc).

The function should use one parameter – to be used to reference the department name.

The return type should be appropriate for returning DepartmentID. The function should return the DepartmentID of the Department that is found. If it is not found, the function should return NULL.

## Requirement 4 – Intermediate Stored Procedure ( 4 Marks )

Create a stored procedure that will insert a record into dbo.Employees. The procedure should accept the following parameters (& the “values” shown beside each field to test it):

* DepartmentName : Infrastructure
* EmployeeFirstName : YourActual FirstName
* EmployeeLastName : YourActual LastName
* Salary
* FileFolder : FirstNameLastName
* ManagerFirstName : James
* ManagerLastName : Donoghue
* CommissionBonus

The Salary parameter should be optional. If not specified, it should default to 42000. The CommissionBonus parameter should be optional. If not specified it should default to 3200.

The procedure should use the function created in requirement 3 to look up the department by the department Name. If the function returns null, a new department should be created.

The procedure should use the function provided in the Project3\_InitialDB.SQL to look up the manager’s employee ID by first name and last name. If the function returns null, a new manager should be created.

The procedure must be tested to insert a new employee, using the DepartmentName, EmployeeFirstName, EmployeeLastName, Salary, FileFolder and CommissionBonus parameters (replacing any FirstName or LastName fields with your name ... eg. if I was Waldo Donoghue then FirstName = Waldo, LastName = Donoghue). For the ManagerEmployeeID and DepartmentID column, it should use the ID that was either found or created in the step above.

Any new departments or managers created by this procedure should not be committed to the database if the insert for the employee fails.

Also execute a Test of this new procedure with these input parameter values :

* DepartmentName : DatabaseMgmt
* EmployeeFirstName : Wherewolf
* EmployeeLastName : Waldo
* Salary : 46000
* FileFolder : FirstNameLastName
* ManagerFirstName : Bill
* ManagerLastName : Gates
* CommissionBonus : 2300

## Requirement 5 – Table Value Function ( 3 Marks )

Write a table value function that will return a table displaying all the employee and department data (without the ID values though) for employees greater than a given salary value. (Only do this if the salary is >= 0, but don’t worry about an error message in this case...). Test this function with a salary value of 44000.

## Requirement 6 – Window Function ( 4 Marks )

Write a window function that will rank employees by department, based on descending CommissionBonus (i.e. highest Commission should be #1). The query should also get the name and Commission of the person above them (ie. So they know who to ask for sales tips to increase their commission). Also include the average commission that shows how each person and department compares to each other. Also add a TotalCompensation column that shows the total of Salary + CommissionBonus. Execute this Window Function to test it.

## Requirement 7 – Recursive CTE ( 5 Marks )

Write a recursive CTE that will get employees by their manager. Include the following columns:

* Employee FirstName
* Employee LastName
* Department ID
* FileFolder
* Manager FirstName
* Manager LastName

The field called FileFolder is used to store the performance review for each employee. Note since managers will also have access to all the people that report to them either directly or indirectly, each manager’s folder will eventually be setup to contain not just their own performance review files, but also all the subfolders for each of the employees that report directly to them. To help facilitate that, also include a column called “File Path” that will determine and show the file path name for each employee using Windows style of \ between subfolders, ie. in the format ManagerFileFolder\EmployeeFileFolder\ etc.