Project 3 INFO5052 Advanced Databases

Project Info

Due Date

Friday, November 8th, 11:59PM

Submission

Submit **a single ".sql"** file (**and a copy in a .txt file**) to the Project 3 submission drop box.

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

Keep working with the same assignment group. Significant overlap between submissions from two or more students may be flagged for plagiarism.

Grading

The project is graded out of 20 total marks. 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. Note that the WHOLE script/solution you submit must run successfully without error, and errors in your submission will incur penalty deductions up to 20% of this project.

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 ALL the ones requested below) . Your whole SQL script must execute without error against the database created by "Project3_InitialDB.sql". Please do NOT submit a SQL server generated script ... instead copy and paste your complete SQL code (including at the top the InitialDB.sql code you started with – so it will run successfully when I submit it…) into a file and submit that (along with the Project3.txt copy) ...

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, thereby testing this stored procedure by creating only the following four departments:

DepartmentName	DepartmentDesc
QA	Quality Assurance
SysDev	Systems Development
Infrastructure	Deployment and Production Support
DesignEngineering	Project Initiation/Design/Engineering

Include 1 Select * SQL statement to confirm the records were successfully added to the correct table.

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. Don't test it yet…

Requirement 4 – Intermediate Stored Procedure (5 Marks)

Create a stored procedure called Insert`Employee` that will insert a record into dbo.Employees. The procedure should accept the following parameters :

- DepartmentName : Deployment
- EmployeeFirstName : Wherewolf
- EmployeeLastName : Waldo
- Salary
- FileFolder : FirstNameLastName
- ManagerFirstName : YourActualFirstName
- ManagerLastName : YourActualLastName
- CommissionBonus

The Salary parameter should be optional. If not specified, it should default to 46000. The CommissionBonus parameter should be optional. If not specified it should default to 5000.

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. When creating new managers, Add $12000 to the default salary for an employee to set their starting salary.

The FileFolder field should be named based on concatenating the Employee First Name and Employee Last Name together, so that it can be used to store & find information for a specific employee.

The procedure must be tested to insert a new employee, using the DepartmentName, EmployeeFirstName, EmployeeLastName, Salary, FileFolder, Manager Names and CommissionBonus parameters (replacing the 1st test ManagerFirstName and ManagerLastName fields with your name ... eg. if I was Waldo Donoghue then ManagerFirstName = Waldo, ManagerLastName = Donoghue). For the ManagerEmployeeID and DepartmentID column, it should use the ID that was either found or created in the step above.

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

Execute 2 Tests of this new procedure with these input parameter values, and also include a simple Select * of all records in the Employees and Departments tables after the 2 tests so you can make sure updates are successful:

Test 1:

- DepartmentName : Deployment
- EmployeeFirstName : Wherewolf
- EmployeeLastName : Waldo
- FileFolder : FirstNameLastName
- ManagerFirstName : YourActualFirstName
- ManagerLastName : YourActualLastName

Test2:

- DepartmentName : Database
- EmployeeFirstName : YourFriendFirstName
- EmployeeLastName : YourFriendLastName
- Salary : 43000
- FileFolder : YourFriendFirstNameYourFriendLastName
- ManagerFirstName : Sarah
- ManagerLastName : Campbell
- CommissionBonus : 4000

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 commission value. (Only execute the select if the commission is >= 0, but don't worry about an error message in this case...). Test this function with a commission value of 5000, AND then again with a commission value of 4000.

Requirement 6 – Window Function (4 Marks)

Write a window function that will rank employees by department, based on descending Compensation (i.e. highest Salary plus commission should be #1). The query should also get the name and compensation of the person above them.

Also include the average compensation that shows how each person and department compares to each other, so that HR can decide how to manage salary and commission levels going forward. Also add a Salary and a Commission column, so users can confirm how the compensation was determined. Execute this Window Function to test it.

Requirement 7 – Recursive CTE (4 Marks)

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

- Employee LastName
- Employee FirstName
- Department ID
- FileFolder
- Manager LastName
- Manager FirstName

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. To illustrate how this would work, if for example I reported directly to Dev Sainani and Dev reports to Peter Devlin, then that File Path would be PeterDevlin\DevSainani\JamesDonoghue