Exercises: Functions and Procedures

This document defines the exercise assignments for the "Databases Basics - MSSQL" course @ Software University.

Part I – Queries for SoftUni Database

Problem 1. Employees with Salary Above 35000

Create stored procedure usp_GetEmployeesSalaryAbove35000 that returns all employees' first and last names for whose salary is above 35000.

Example

First Name	Last Name
Roberto	Tamburello
David	Bradley
Terri	Duffy

Problem 2. Employees with Salary Above Number

Create stored procedure usp_GetEmployeesSalaryAboveNumber that accept a number (of type DECIMAL(18,4)) as parameter and returns all employees' first and last names whose salary is above or equal to the given number.

Example

Supplied number for that example is 48100.

First Name	Last Name	
Terri	Duffy	
Jean	Trenary	
Ken	Sanchez	
	•••	

Problem 3. Town Names Starting With

Write a stored procedure usp_GetTownsStartingWith that accept string as parameter and returns all town names starting with that string.

Example

Here is the list of all towns starting with "b".

Town
Bellevue
Bothell
Bordeaux
Berlin

















Problem 4. Employees from Town

Write a stored procedure usp_GetEmployeesFromTown that accepts town name as parameter and return the employees' first and last name that live in the given town.

Example

Here it is a list of employees living in Sofia.

First Name	Last Name	
Svetlin	Nakov	
Martin	Kulov	
George	Denchev	

Problem 5. Salary Level Function

Write a function ufn_GetSalaryLevel(@salary DECIMAL(18,4)) that receives salary of an employee and returns the level of the salary.

- If salary is < 30000 return "Low"
- If salary is between 30000 and 50000 (inclusive) return "Average"
- If salary is > 50000 return "High"

Example

Salary	Salary Level
13500.00	Low
43300.00	Average
125500.00	High

Problem 6. Employees by Salary Level

Write a stored procedure usp_EmployeesBySalaryLevel that receive as parameter level of salary (low, average or high) and print the names of all employees that have given level of salary. You should use the function -"dbo.ufn_GetSalaryLevel(@Salary)", which was part of the previous task, inside your "CREATE PROCEDURE ..." query.

Example

Here is the list of all employees with high salary.

First Name	Last Name
Terri	Duffy
Jean	Trenary
Ken	Sanchez

Problem 7. Define Function

Define a function ufn_IsWordComprised(@setOfLetters, @word) that returns true or false depending on that if the word is a comprised of the given set of letters.

















Example

SetOfLetters	Word	Result
oistmiahf	Sofia	1
oistmiahf	halves	0
bobr	Rob	1
рррр	Guy	0

Problem 8. * Delete Employees and Departments

Write a procedure with the name usp_DeleteEmployeesFromDepartment (@departmentId INT) which deletes all Employees from a given department. Delete these departments from the Departments table too. Finally SELECT the number of employees from the given department. If the delete statements are correct the select query should return 0.

After completing that exercise restore your database to revert all changes.

Hint:

You may set ManagerID column in Departments table to nullable (using query "ALTER TABLE ...").

PART II – Queries for Bank Database

Find Full Name Problem 9.

You are given a database schema with tables AccountHolders(Id (PK), FirstName, LastName, SSN) and Accounts(Id (PK), AccountHolderId (FK), Balance). Write a stored procedure usp_GetHoldersFullName that selects the full names of all people.

Example

Full Name
Susan Cane
Kim Novac
Jimmy Henderson

Problem 10. People with Balance Higher Than

Your task is to create a stored procedure usp GetHoldersWithBalanceHigherThan that accepts a number as a parameter and returns all people who have more money in total of all their accounts than the supplied number.

Example

First Name	Last Name	
Susan	Cane	
Petar	Kirilov	

















Problem 11. Future Value Function

Your task is to create a function ufn_CalculateFutureValue that accepts as parameters – sum (decimal), yearly interest rate (float) and number of years(int). It should calculate and return the future value of the initial sum. Using the following formula:

$$FV = I \times ((1+R)^T)$$

- I Initial sum
- **R** Yearly interest rate
- T Number of years

Example

Input	Output
Initial sum: 1000	1610.51
Yearly Interest rate: 10%	
years: 5	
ufn_CalculateFutureValue(1000, 0.1, 5)	

Problem 12. Calculating Interest

Your task is to create a stored procedure usp_CalculateFutureValueForAccount that uses the function from the previous problem to give an interest to a person's account for 5 years, along with information about his/her account id, first name, last name and current balance as it is shown in the example below. It should take the AccountId and the interest rate as parameters. Again you are provided with "dbo.ufn_CalculateFutureValue" function which was part of the previous task.

Example

Account Id	First Name	Last Name	Current Balance	Balance in 5 years
1	Susan	Cane	123.12	198.286

^{*}Note: for the example above interest rate is 0.1

PART III – Queries for Diablo Database

You are given a database "Diablo" holding users, games, items, characters and statistics available as SQL script. Your task is to write some stored procedures, views and other server-side database objects and write some SQL queries for displaying data from the database.

Important: start with a clean copy of the "Diablo" database on each problem. Just execute the SQL script again.

Problem 13. *Scalar Function: Cash in User Games Odd Rows

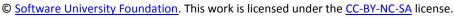
Create a function ufn_CashInUsersGames that sums the cash of odd rows. Rows must be ordered by cash in descending order. The function should take a game name as a parameter and return the result as table. Submit only your function in.

Execute the function over the following game names, ordered exactly like: "Lily Stargazer", "Love in a mist".

Output

SumCash 5515.00



















7266.00

Hint

Use **ROW_NUMBER** to get the rankings of all rows based on order criteria.















