* + **Всичко от точка 1-6 може да бъде намерено в Accounts.sql.**
  + **Точка 7 в TelerikAcademy.sql, StoredProcedures за FindPersonAndTowns и Functions-Scalar-valued Functions за TextContainingLetter**
  + **Точка 8 е във ScanCursor.sql**
  + **За точка 9 може от тук** [**http://www.mssqltips.com/sqlservertip/2022/concat-aggregates-sql-server-clr-function/**](http://www.mssqltips.com/sqlservertip/2022/concat-aggregates-sql-server-clr-function/) **да се види как да се създаде самия Dll**

1. **Create a database with two tables: Persons(Id(PK), FirstName, LastName, SSN) and Accounts(Id(PK), PersonId(FK), Balance). Insert few records for testing. Write a stored procedure that selects the full names of all persons.**

USE Accounts

GO

CREATE PROC dbo.usp\_ShowFullName

AS

Select FirstName + ' ' + LastName as [Full name] from Persons

GO

1. **Create a stored procedure that accepts a number as a parameter and returns all persons who have more money in their accounts than the supplied number.**

USE Accounts

GO

CREATE PROC dbo.usp\_SelectPersonsByBalance(@balance money)

AS

Select p.FirstName + ' ' + p.LastName as [Full name] from Persons p

inner join Accounts a on p.PersonId=a.PersonId

Where a.Balance>@balance

GO

1. **Create a function that accepts as parameters – sum, yearly interest rate and number of months. It should calculate and return the new sum. Write a SELECT to test whether the function works as expected.**

USE Accounts

GO

CREATE FUNCTION ufn\_CalcInterest(@sum money, @yearlyInterestRate float, @months int)

RETURNS money

AS

BEGIN

return ( @sum \* @yearlyInterestRate \* 12/@months )

END

USE [Accounts]

GO

SELECT [dbo].[ufn\_CalcInterest] (100, 4, 6)

GO

1. **Create a stored procedure that uses the function from the previous example to give an interest to a person's account for one month. It should take the AccountId and the interest rate as parameters.**

USE [Accounts]

GO

CREATE PROC dbo.usp\_GiveInterest(@Account int, @InterestRate float)

AS

Update Accounts

Set Balance = [dbo].[ufn\_CalcInterest] (Balance, @InterestRate, 1)

where PersonId = @Account

GO

1. **Add two more stored procedures WithdrawMoney( AccountId, money) and DepositMoney (AccountId, money) that operate in transactions.**

USE [Accounts]

GO

CREATE PROC dbo.usp\_WithdrawMoney(@Account int, @money money)

AS

BEGIN TRAN

Update Accounts

Set Balance = Balance - @money

where PersonId = @Account

COMMIT TRAN

GO

USE [Accounts]

GO

CREATE PROC dbo.usp\_DepositMoney(@Account int, @money money)

AS

BEGIN TRAN

Update Accounts

Set Balance = Balance + @money

where PersonId = @Account

COMMIT TRAN

GO

1. **Create another table – Logs(LogID, AccountID, OldSum, NewSum). Add a trigger to the Accounts table that enters a new entry into the Logs table every time the sum on an account changes.**

USE [Accounts]

GO

CREATE TRIGGER tr\_BalanceChanged ON Accounts FOR INSERT, UPDATE, DELETE

AS

Begin

Insert into Logs(AccountId, OldSum, NewSum) Select i.AccountID, i.Balance, d.Balance from inserted i, deleted d

End;

GO

1. **Define a function in the database TelerikAcademy that returns all Employee's names (first or middle or last name) and all town's names that are comprised of given set of letters. Example 'oistmiahf' will return 'Sofia', 'Smith', … but not 'Rob' and 'Guy'.**

USE [TelerikAcademy]

GO

CREATE FUNCTION fn\_TextContainingLetters(

@text nvarchar(100),

@letters nvarchar(100)

)

Returns nvarchar

AS

BEGIN

DECLARE @result nvarchar = 'T'

DECLARE @letter nvarchar(1)

DECLARE @counter int = 1

While(@counter <= LEN(@text))

Begin

SET @letter = SUBSTRING(@text, @counter, 1)

IF (CHARINDEX(@letter, @letters) = 0)

SET @result = 'F'

SET @counter = @counter + 1

End

Return @result

End

USE [TelerikAcademy]

GO

CREATE PROC usp\_FindPersonAndTownsName(@lettersToSearch NVARCHAR(50))

AS

Select FirstName from Employees

where (Select [dbo].[fn\_TextContainingLetters](FirstName, @lettersToSearch)) = 'T'

UNION

Select MiddleName from Employees

where (Select [dbo].[fn\_TextContainingLetters](MiddleName, @lettersToSearch)) = 'T'

UNION

Select LastName from Employees

where (Select [dbo].[fn\_TextContainingLetters](LastName, @lettersToSearch)) = 'T'

UNION

Select Name from Towns

where (Select [dbo].[fn\_TextContainingLetters](Name, @lettersToSearch)) = 'T'

GO

1. **Using database cursor write a T-SQL script that scans all employees and their addresses and prints all pairs of employees that live in the same town.**

USE [TelerikAcademy]

GO

DECLARE ScanCursor CURSOR READ\_ONLY FOR

SELECT e.FirstName, e.LastName, t.Name, o.FirstName, o.LastName

FROM Employees e

INNER JOIN Addresses a ON a.AddressID = e.AddressID

INNER JOIN Towns t ON t.TownID = a.TownID,

Employees o

INNER JOIN Addresses ad ON ad.AddressID = o.AddressID

INNER JOIN Towns tw ON tw.TownID = ad.TownID

OPEN ScanCursor

DECLARE @EmployeeOneFirstName NVARCHAR(20)

DECLARE @EmployeeOneLastName NVARCHAR(30)

DECLARE @EmployeeTwoFirstName NVARCHAR(20)

DECLARE @EmployeeTwoLastName NVARCHAR(30)

DECLARE @town NVARCHAR(30)

FETCH NEXT FROM ScanCursor INTO @EmployeeOneFirstName, @EmployeeOneLastName, @town, @EmployeeTwoFirstName, @EmployeeTwoLastName

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT @EmployeeOneFirstName + ' ' + @EmployeeOneLastName +'--' + @town + '--' + @EmployeeTwoFirstName + ' ' + @EmployeeTwoLastName

FETCH NEXT FROM ScanCursor INTO @EmployeeOneFirstName, @EmployeeOneLastName, @town, @EmployeeTwoFirstName, @EmployeeTwoLastName

END

CLOSE ScanCursor

DEALLOCATE ScanCursor

1. **Define a .NET aggregate function StrConcat that takes as input a sequence of strings and return a single string that consists of the input strings separated by ','. For example the following SQL statement should return a single string:**

**SELECT StrConcat(FirstName + ' ' + LastName)**

**FROM Employees**

USE [TelerikAcademy]

GO

CREATE Assembly concatAssembly

AUTHORIZATION dbo

FROM 'D:\Concatination.dll'

WITH PERMISSION\_SET = SAFE;

GO

CREATE AGGREGATE StrConcat(@input nvarchar)

RETURNS nvarchar(4000)

EXTERNAL NAME concatAssembly.StrConcat;

GO