## Transact SQL Homework

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.**

CREATE DATABASE BankSystem

GO

CREATE TABLE Persons(

PersonID int IDENTITY PRIMARY KEY,

FirstName nvarchar(50) NOT NULL,

LastName nvarchar(50) NOT NULL,

SSN int NOT NULL UNIQUE

)

GO

CREATE TABLE Accounts(

AccountID int IDENTITY PRIMARY KEY,

PersonID int NOT NULL,

Balance money NOT NULL,

CONSTRAINT FK\_Accounts\_PErsons

FOREIGN KEY(PersonID)

REFERENCES Persons(PersonID)

)

GO

INSERT INTO Persons(FirstName, LastName, SSN)

VALUES('Ivaylo', 'Kenov', 123456),

('Nikolay', 'Kostov', 234567),

('Donsho', 'Minkov', 345678),

('Evlogi', 'Hristov', 456789)

GO

INSERT INTO Accounts(PersonID, Balance)

VALUES(4, 2000),

(1, 1500),

(3,6200),

(2, 380)

GO

CREATE PROC usp\_SelectFullNamesOfPersons

AS

SELECT FirstName + ' ' + LastName AS [Full Name]

FROM Persons

GO

EXEC usp\_SelectFullNamesOfPersons

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.**

CREATE PROC usp\_GetPersonsByBalance(@minimalBalance money = 0)

AS

SELECT p.FirstName + ' ' + p.LastName AS [Full Name], a.Balance

FROM Persons p

JOIN Accounts a

ON p.PersonID = a.PersonID

WHERE a.Balance > @minimalBalance

GO

EXEC usp\_GetPersonsByBalance 1600

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.**

CREATE FUNCTION dbo.ufn\_CalculateInterest(@sum money, @yearInterestRate decimal, @months decimal)

RETURNS money

AS

BEGIN

RETURN @sum + (@months/12) \* (@yearInterestRate/100) \* @sum

END

GO

SELECT dbo.ufn\_CalculateInterest (1000, 6, 24) AS Interest

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.**

CREATE PROC usp\_SelectPersonsAndCalculateInterestForMonth(@accountID int, @interestRate decimal)

AS

UPDATE Accounts

SET Balance = dbo.ufn\_CalculateInterest(Balance, @interestRate, 1)

WHERE AccountID = @accountID

GO

EXEC usp\_SelectPersonsAndCalculateInterestForMonth 1, 10

GO

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

CREATE PROC usp\_WithdrawMoney(@accountID int, @moneyToWithdraw money)

AS

UPDATE Accounts

SET Balance = (Balance - @moneyToWithdraw)

WHERE @accountID = AccountID

GO

usp\_WithdrawMoney 2, 100

GO

CREATE PROC usp\_DepositMoney(@accountID int, @moneyToWithdraw money)

AS

UPDATE Accounts

SET Balance = (Balance + @moneyToWithdraw)

WHERE @accountID = AccountID

GO

usp\_DepositMoney 2, 400

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.**

CREATE TABLE Logs(

LogID int IDENTITY PRIMARY KEY,

AccountID int NOT NULL,

OldSum money,

NewSum money NOT NULL,

CONSTRAINT FK\_Logs\_Accounts

FOREIGN KEY (AccountID)

REFERENCES Accounts(AccountID)

)

GO

CREATE TRIGGER tr\_AccountsUpdare ON Accounts

FOR UPDATE

AS

INSERT INTO Logs(AccountID, OldSum, NewSum)

SELECT d.AccountID, d.Balance, i.Balance

FROM deleted d

JOIN inserted i on d.AccountID = i.AccountID

GO

UPDATE Accounts

SET Balance = 2222

WHERE AccountID = 2

GO

CREATE TRIGGER tr\_AccountsInsert ON Accounts

FOR INSERT

AS

INSERT INTO Logs(AccountID, OldSum, NewSum)

SELECT i.AccountID, NULL, i.Balance

FROM inserted i

GO

INSERT INTO Persons(FirstName, LastName, SSN)

VALUES('Antony', 'Zhekov', 987654)

INSERT INTO Accounts(PersonID, Balance)

VALUES(5, 3333)

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'.**

No time, so I decided not to copy the other tasks from the forum.

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.**
2. **\* Write a T-SQL script that shows for each town a list of all employees that live in it. Sample output:**
3. **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:**