07. DB-Basics-Database-Programmability-and-Transactions-Exercises

/\* 01EmployeesWithSalaryAbove35000\*/

CREATE PROC usp\_GetEmployeesSalaryAbove35000

AS

SELECT FirstName AS [First Name], LastName AS [Last Name]

FROM Employees

WHERE Salary > 35000

--In Judge must be paste without this below

GO

EXEC usp\_GetEmployeesSalaryAbove35000

/\* 02EmployeesWithSalaryAboveNumber\*/

CREATE PROC usp\_GetEmployeesSalaryAboveNumber(@MinSalary DECIMAL(18, 4))

AS

SELECT FirstName AS [First Name], LastName AS [Last Name]

FROM Employees

WHERE Salary >= @MinSalary

--In Judge must be paste without this below

GO

EXEC usp\_GetEmployeesSalaryAboveNumber 48100

/\* 03TownNamesStartingWith\*/

CREATE PROC usp\_GetTownsStartingWith(@StartingPattern VARCHAR(50)) AS

SELECT [Name] AS Town

FROM Towns

WHERE [Name] LIKE @StartingPattern + '%'

--In Judge must be paste without this below

GO

EXEC usp\_GetTownsStartingWith 'B'

/\* 04EmployeesFromTown\*/

CREATE PROCEDURE usp\_GetEmployeesFromTown(@TownName VARCHAR(50)) AS

SELECT e.FirstName AS [First Name], e.LastName AS [Last Name]

FROM Employees AS e

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

JOIN Towns AS t ON t.TownID = a.TownID

WHERE t.[Name] = @TownName

--Only one query must be paste in Judge

GO

DROP PROC usp\_GetEmployeesFromTown

GO

CREATE PROCEDURE usp\_GetEmployeesFromTown(@TownName VARCHAR(50)) AS

SELECT e.FirstName AS [First Name], e.LastName AS [Last Name]

FROM Employees AS e

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

JOIN Towns AS t ON t.TownID = a.TownID

WHERE t.[Name] LIKE @TownName + '%'

--In Judge must be paste without this below

GO

EXEC usp\_GetEmployeesFromTown 'Sofia'

/\* 05SalaryLevelFunction\*/

CREATE FUNCTION ufn\_GetSalaryLevel(@salary DECIMAL(18, 4))

RETURNS CHAR(7) AS

BEGIN

DECLARE @salaryLevel CHAR(7)

IF (@salary < 30000)

SET @salaryLevel = 'Low'

ELSE IF (@salary BETWEEN 30000 AND 50000)

SET @salaryLevel = 'Average'

ELSE

SET @salaryLevel = 'High'

RETURN @salaryLevel

END

--In Judge must be paste without this below

GO

SELECT e.Salary, dbo.ufn\_GetSalaryLevel(e.Salary) AS [Salary Level]

FROM Employees AS e

/\* 06EmployeesBySalaryLevel\*/

CREATE PROCEDURE usp\_EmployeesBySalaryLevel (@salaryLevel CHAR(7)) AS

SELECT FirstName AS [First Name], LastName AS [Last Name]

FROM Employees

WHERE dbo.ufn\_GetSalaryLevel(Salary) = @salaryLevel

--In Judge must be paste without this below

GO

EXEC usp\_EmployeesBySalaryLevel 'High'

/\* 07DefineFunction\*/

CREATE FUNCTION ufn\_IsWordComprised(@setOfLetters VARCHAR(MAX), @word VARCHAR(MAX))

RETURNS BIT AS

BEGIN

DECLARE @result BIT = 0

DECLARE @isCounted BIT = 0

DECLARE @wordIndex INT = 1

DECLARE @setOfLettersIndex INT = 1

DECLARE @counter INT = 0

WHILE(@wordIndex <= LEN(@word))

BEGIN

WHILE(@setOfLettersIndex <= LEN(@setOfLetters))

BEGIN

IF (SUBSTRING(@word, @wordIndex, 1) = SUBSTRING(@setOfLetters, @setOfLettersIndex, 1)

AND @isCounted = 0)

BEGIN

SET @counter += 1

SET @isCounted = 1

END

SET @setOfLettersIndex += 1

END

SET @isCounted = 0

SET @wordIndex += 1

SET @setOfLettersIndex = 1

END

IF (@counter = LEN(@word))

SET @result = 1

RETURN @result

END

--Only one query must be paste in Judge

GO

CREATE FUNCTION ufn\_IsWordComprised(@setOfLetters VARCHAR(MAX), @word VARCHAR(MAX))

RETURNS BIT AS

BEGIN

DECLARE @index INT = 1

WHILE(@index <= LEN(@word))

BEGIN

DECLARE @letter CHAR(1) = SUBSTRING(@word, @index, 1)

IF (CHARINDEX(@letter, @setOfLetters) <= 0)

BEGIN

RETURN 0

END

SET @index += 1

END

RETURN 1

END

--In Judge must be paste without this below

GO

DROP FUNCTION dbo.ufn\_IsWordComprised

GO

SELECT dbo.ufn\_IsWordComprised('gug', 'Guy') AS Result

/\* 08DeleteEmployeesAndDepartments\*/

CREATE PROCEDURE usp\_DeleteEmployeesFromDepartment(@departmentId INT) AS

BEGIN

ALTER TABLE Employees ALTER COLUMN ManagerID INT

ALTER TABLE Employees ALTER COLUMN DepartmentID INT

UPDATE Employees

SET DepartmentID = NULL

WHERE DepartmentID = @departmentId

--WHERE EmployeeID IN (SELECT EmployeeID FROM Employees WHERE DepartmentID = @departmentId)

UPDATE Employees

SET ManagerID = NULL

WHERE ManagerID IN (SELECT EmployeeID FROM Employees WHERE DepartmentID = @departmentId)

ALTER TABLE Departments ALTER COLUMN ManagerID INT

UPDATE Departments

SET ManagerID = NULL WHERE DepartmentID = @departmentId

DELETE FROM Departments WHERE DepartmentID = @departmentId

DELETE FROM EmployeesProjects

WHERE EmployeeID IN (SELECT EmployeeID FROM Employees WHERE DepartmentID = @departmentId)

DELETE FROM Employees WHERE DepartmentID = @departmentId

SELECT COUNT(\*) FROM Employees WHERE DepartmentID = @departmentId

END

--In Judge must be paste without this below

EXEC usp\_DeleteEmployeesFromDepartment 3

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\* NOT FOR JUDGE \*\*\*\*\* NOT FOR JUDGE \*\*\*\*\*

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Understood from the old exercise explanation

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem 8.1. \* Delete Employees and Departments

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE PROCEDURE usp\_DeleteEmployeesFromDepartment

(

@departmentName NVARCHAR(50)

)

AS

BEGIN

ALTER TABLE Employees ALTER COLUMN ManagerID INT;

ALTER TABLE Employees ALTER COLUMN DepartmentID INT;

UPDATE Employees

SET

DepartmentID = NULL

WHERE EmployeeID IN

(

(

SELECT e.EmployeeID

FROM Employees AS e

JOIN Departments AS d ON e.DepartmentID = d.DepartmentID

WHERE d.Name = @departmentName

)

);

UPDATE Employees

SET

ManagerID = NULL

WHERE ManagerID IN

(

(

SELECT e.EmployeeID

FROM Employees AS e

JOIN Departments AS d ON e.DepartmentID = d.DepartmentID

WHERE d.Name = @departmentName

)

);

ALTER TABLE Departments ALTER COLUMN ManagerID INT;

UPDATE Departments

SET

ManagerID = NULL

WHERE Name = @departmentName;

DELETE FROM Departments

WHERE Name = @departmentName;

DELETE FROM EmployeesProjects

WHERE EmployeeID IN

(

(

SELECT e.EmployeeID

FROM Employees AS e

JOIN Departments AS d ON e.DepartmentID = d.DepartmentID

WHERE d.Name = @departmentName

)

);

DELETE FROM Employees

WHERE EmployeeID IN

(

(

SELECT e.EmployeeID

FROM Employees AS e

JOIN Departments AS d ON e.DepartmentID = d.DepartmentID

WHERE d.Name = @departmentName

)

);

END;

BEGIN TRANSACTION;

EXECUTE usp\_DeleteEmployeesFromDepartment

'Production';

EXECUTE usp\_DeleteEmployeesFromDepartment

'Production Control';

ROLLBACK;

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\* NOT FOR JUDGE \*\*\*\*\* NOT FOR JUDGE \*\*\*\*\*

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Understood from the old exercise explanation

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Option without procedure

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ALTER TABLE Employees ALTER COLUMN ManagerID INT;

ALTER TABLE Employees ALTER COLUMN DepartmentID INT;

UPDATE Employees

SET

DepartmentID = NULL

WHERE EmployeeID IN

(

(

SELECT e.EmployeeID

FROM Employees AS e

JOIN Departments AS d ON e.DepartmentID = d.DepartmentID

WHERE d.Name IN('Production', 'Production Control')

)

);

UPDATE Employees

SET

ManagerID = NULL

WHERE ManagerID IN

(

(

SELECT e.EmployeeID

FROM Employees AS e

JOIN Departments AS d ON e.DepartmentID = d.DepartmentID

WHERE d.Name IN('Production', 'Production Control')

)

);

ALTER TABLE Departments ALTER COLUMN ManagerID INT;

UPDATE Departments

SET

ManagerID = NULL

WHERE Name IN('Production', 'Production Control');

DELETE FROM Departments

WHERE Name IN('Production', 'Production Control');

DELETE FROM EmployeesProjects

WHERE EmployeeID IN

(

(

SELECT e.EmployeeID

FROM Employees AS e

JOIN Departments AS d ON e.DepartmentID = d.DepartmentID

WHERE d.Name IN('Production', 'Production Control')

)

);

DELETE FROM Employees

WHERE EmployeeID IN

(

(

SELECT e.EmployeeID

FROM Employees AS e

JOIN Departments AS d ON e.DepartmentID = d.DepartmentID

WHERE d.Name IN('Production', 'Production Control')

)

);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\* FOR JUDGE \*\*\*\*\* FOR JUDGE \*\*\*\*\* FOR JUDGE \*\*\*\*\*

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NEW exercise explanation

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Problem 8.2. \* Delete Employees and Departments

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE PROCEDURE usp\_DeleteEmployeesFromDepartment

(

@departmentId INT

)

AS

BEGIN

ALTER TABLE Employees ALTER COLUMN ManagerID INT;

ALTER TABLE Employees ALTER COLUMN DepartmentID INT;

UPDATE Employees

SET

DepartmentID = NULL

WHERE EmployeeID IN

(

(

SELECT EmployeeID

FROM Employees

WHERE DepartmentID = @departmentId

)

);

UPDATE Employees

SET

ManagerID = NULL

WHERE ManagerID IN

(

(

SELECT EmployeeID

FROM Employees

WHERE DepartmentID = @departmentId

)

);

ALTER TABLE Departments ALTER COLUMN ManagerID INT;

UPDATE Departments

SET

ManagerID = NULL

WHERE DepartmentID = @departmentId;

DELETE FROM Departments

WHERE DepartmentID = @departmentId;

DELETE FROM EmployeesProjects

WHERE EmployeeID IN

(

(

SELECT EmployeeID

FROM Employees

WHERE DepartmentID = @departmentId

)

);

DELETE FROM Employees

WHERE DepartmentID = @departmentId;

SELECT COUNT(\*)

FROM Employees

WHERE DepartmentID = @departmentId;

END;

/\* 09FindFullName\*/

CREATE PROCEDURE usp\_GetHoldersFullName AS

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

FROM AccountHolders

--In Judge must be paste without this below

GO

EXECUTE usp\_GetHoldersFullName

/\* 10PeopleWithBalanceHigherThan\*/

CREATE PROCEDURE usp\_GetHoldersWithBalanceHigherThan(@balance MONEY) AS

SELECT ah.FirstName AS [First Name], ah.LastName AS [Last Name]

FROM AccountHolders AS ah

JOIN Accounts AS a ON a.AccountHolderId = ah.Id

GROUP BY ah.FirstName, ah.LastName

HAVING SUM(a.Balance) > @balance

--In Judge must be paste without this below

GO

EXECUTE usp\_GetHoldersWithBalanceHigherThan 1234.5667

/\* 11FutureValueFunction\*/

CREATE FUNCTION ufn\_CalculateFutureValue(@InitialSum DECIMAL(15, 4), @YearlyInterestRate FLOAT,

@NumberOfYears INT)

RETURNS DECIMAL(15, 4)

AS

BEGIN

DECLARE @futureValue DECIMAL(15, 4)

SET @futureValue = @InitialSum \* (POWER((1 + @YearlyInterestRate), @NumberOfYears))

RETURN @futureValue

END

--In Judge must be paste without this below

GO

SELECT dbo.ufn\_CalculateFutureValue(1000, 0.1, 5)

/\* 12CalculatingInterest\*/

CREATE PROCEDURE usp\_CalculateFutureValueForAccount(@AccountId INT, @YearlyInterestRate FLOAT) AS

BEGIN

SELECT a.Id AS [Account Id], ah.FirstName AS [First Name], ah.LastName AS [Last Name],

a.Balance AS [Current balance],

dbo.ufn\_CalculateFutureValue(a.Balance, @YearlyInterestRate, 5) AS [Balance in 5 years]

FROM Accounts AS a

JOIN AccountHolders AS ah ON ah.Id = a.AccountHolderId

WHERE a.Id = @AccountId

END

--In Judge must be paste without this below

GO

EXECUTE dbo.usp\_CalculateFutureValueForAccount 1, 0.1

/\* 13ScalarFunctionCashInUserGamesOddRows\*/

CREATE FUNCTION ufn\_CashInUsersGames(@GameName NVARCHAR(50))

RETURNS TABLE AS

RETURN

(

SELECT SUM(ncr.Cash) AS SumCash

FROM

(

SELECT g.[Name], ug.Cash, ROW\_NUMBER() OVER(ORDER BY ug.Cash DESC) AS RowNumber

FROM Games AS g

JOIN UsersGames AS ug ON ug.GameId = g.Id

WHERE g.[Name] = @GameName

) AS ncr

WHERE ncr.RowNumber % 2 != 0

)

--In Judge must be paste without this below

GO

SELECT \* FROM ufn\_CashInUsersGames('Lily Stargazer')

SELECT \* FROM ufn\_CashInUsersGames('Love in a mist')

DROP FUNCTION ufn\_CashInUsersGames

/\* 14CreateTableLogs\*/

CREATE TABLE Logs

(

LogId INT NOT NULL IDENTITY CONSTRAINT PK\_Logs PRIMARY KEY,

AccountId INT NOT NULL CONSTRAINT FK\_Logs\_Accounts FOREIGN KEY REFERENCES Accounts(Id),

OldSum MONEY NOT NULL,

NewSum MONEY NOT NULL

)

GO

--In Judge must be paste only the Trigger's creation below

CREATE TRIGGER tr\_Accounts\_Logs\_After\_Update ON Accounts

FOR UPDATE AS

BEGIN

INSERT INTO Logs

VALUES

(

(SELECT Id FROM deleted),

(SELECT Balance FROM deleted),

(SELECT Balance FROM inserted)

)

END

--In Judge must be paste without this below

GO

UPDATE Accounts SET Balance -= 10 WHERE Id = 1

UPDATE Accounts SET Balance += 10 WHERE Id = 1

SELECT \* FROM Logs

/\* 15CreateTableEmails\*/

CREATE TABLE NotificationEmails

(

Id INT NOT NULL IDENTITY CONSTRAINT PK\_NotificationEmails PRIMARY KEY,

Recipient INT NOT NULL,

[Subject] NVARCHAR(50) NOT NULL,

Body NVARCHAR(300)

)

GO

--In Judge must be paste only the Trigger's creation below

CREATE TRIGGER tr\_Logs\_NotificationEmails\_AfterInsert ON Logs

FOR INSERT AS

BEGIN

INSERT INTO NotificationEmails

VALUES

(

(SELECT AccountId FROM inserted),

CONCAT('Balance change for account: ', (SELECT AccountId FROM inserted)),

CONCAT('On ', FORMAT(GETDATE(), 'MMM dd yyyy HH mm'), ' your balance was changed from ',

(SELECT OldSum FROM Logs), ' to ', (SELECT NewSum FROM Logs), '.')

)

END

--In Judge must be paste without this below

GO

UPDATE Accounts SET Balance -= 10 WHERE Id = 1

UPDATE Accounts SET Balance += 10 WHERE Id = 1

INSERT INTO Logs

VALUES

(1, 123.12, 123.12)

SELECT \* FROM NotificationEmails

DROP TRIGGER tr\_Logs\_NotificationEmails\_AfterInsert

/\* 16DepositMoney\*/

CREATE PROCEDURE usp\_DepositMoney(@AccountId INT, @MoneyAmount MONEY) AS

BEGIN

IF (@MoneyAmount < 0)

BEGIN

RAISERROR('Cannot deposit negative value!', 16, 1)

END

ELSE

BEGIN

IF (@AccountId IS NULL OR @MoneyAmount IS NULL)

RAISERROR('Missing value!', 16, 1)

END

BEGIN TRANSACTION

UPDATE Accounts SET Balance += @MoneyAmount WHERE Id = @AccountId

IF (@@ROWCOUNT <> 1)

BEGIN

ROLLBACK

RAISERROR('Invalid account!', 16, 1)

RETURN

END

COMMIT

END

--In Judge must be paste without this below

GO

EXECUTE usp\_DepositMoney 1, 10

EXECUTE usp\_WithdrawMoney 1, 10

SELECT \* FROM Accounts

/\* 17WithdrawMoney\*/

CREATE PROCEDURE usp\_WithdrawMoney @AccountId INT, @MoneyAmount MONEY AS

BEGIN

IF (@MoneyAmount < 0)

BEGIN

RAISERROR('Cannot deposit negative value!', 16, 1)

END

ELSE

BEGIN

IF (@AccountId IS NULL OR @MoneyAmount IS NULL)

RAISERROR('Missing value!', 16, 1)

END

BEGIN TRANSACTION

UPDATE Accounts SET Balance -= @MoneyAmount WHERE Id = @AccountId

IF (@@ROWCOUNT <> 1)

BEGIN

ROLLBACK

RAISERROR('Invalid account!', 16, 1)

RETURN

END

ELSE

BEGIN

IF ((SELECT Balance FROM Accounts WHERE Id = @AccountId) < 0)

BEGIN

ROLLBACK

RAISERROR('The money in this account are not enough!', 16, 1)

RETURN

END

END

COMMIT

END

--In Judge must be paste without this below

GO

EXECUTE usp\_WithdrawMoney 5, 25

EXECUTE usp\_DepositMoney 5, 25

SELECT \* FROM Accounts

/\* 18MoneyTransfer\*/

CREATE PROCEDURE usp\_TransferMoney(@SenderId INT, @ReceiverId INT, @MoneyAmount MONEY) AS

BEGIN

IF (@MoneyAmount < 0)

BEGIN

RAISERROR('Cannot deposit negative value!', 16, 1)

END

ELSE

BEGIN

IF (@SenderId IS NULL OR @ReceiverId IS NULL OR @MoneyAmount IS NULL)

RAISERROR('Missing value!', 16, 1)

END

BEGIN TRANSACTION

UPDATE Accounts SET Balance -= @MoneyAmount WHERE Id = @SenderId

IF (@@ROWCOUNT <> 1)

BEGIN

ROLLBACK

RAISERROR('Invalid account!', 16, 1)

RETURN

END

IF ((SELECT Balance FROM Accounts WHERE Id = @SenderId) < 0)

BEGIN

ROLLBACK

RAISERROR('The money in this account are not enough!', 16, 1)

RETURN

END

UPDATE Accounts SET Balance += @MoneyAmount WHERE Id = @ReceiverId

IF (@@ROWCOUNT <> 1)

BEGIN

ROLLBACK

RAISERROR('Invalid account!', 16, 1)

RETURN

END

COMMIT

END

--Only one query must be paste in Judge

GO

CREATE PROCEDURE usp\_TransferMoney(@SenderId INT, @ReceiverId INT, @MoneyAmount MONEY) AS

BEGIN

BEGIN TRANSACTION

EXECUTE usp\_WithdrawMoney @SenderId, @MoneyAmount

EXECUTE usp\_DepositMoney @ReceiverId, @MoneyAmount

COMMIT

END

--Only one query must be paste in Judge

GO

CREATE PROCEDURE usp\_TransferMoney(@SenderId INT, @ReceiverId INT, @MoneyAmount MONEY) AS

BEGIN

EXECUTE usp\_WithdrawMoney @SenderId, @MoneyAmount

EXECUTE usp\_DepositMoney @ReceiverId, @MoneyAmount

END

--In Judge must be paste without this below

GO

DROP PROCEDURE usp\_TransferMoney

EXECUTE usp\_TransferMoney 5, 1, 5000

EXECUTE usp\_TransferMoney 1, 5, 5000

SELECT \* FROM Accounts

/\* 19Trigger\*/

--Creating trigger

CREATE TRIGGER tr\_UserGameItems\_LevelRestriction ON UserGameItems

INSTEAD OF INSERT

AS

INSERT INTO UserGameItems

SELECT ItemId, UserGameId FROM inserted

WHERE ItemId IN

(SELECT Id FROM Items WHERE MinLevel <= (SELECT [Level] FROM UsersGames WHERE Id = UserGameId))

--Warning for adding bonus!!!

--The bonus must be added only one time (50000 or more) - using one from updating queries or

--creating and executing procedure usp\_GiveBonusInCash(executing five times with 50000 or with @bonusCash)

--Adding bonus cash as updating table (it must be execute only this query below or usp\_GiveBonusInCash)

UPDATE ug SET ug.Cash += 50000

FROM UsersGames AS ug

JOIN Users AS u ON u.Id = ug.UserId

JOIN Games AS g ON g.Id = ug.GameId

WHERE g.[Name] = 'Bali' AND

u.Username IN('baleremuda', 'loosenoise','inguinalself','buildingdeltoid','monoxidecos')

GO

CREATE PROCEDURE usp\_GiveBonusInCash (@GameName NVARCHAR(50), @Username NVARCHAR(50),

@CashAmount MONEY) AS

BEGIN

UPDATE ug SET ug.Cash += @CashAmount

FROM UsersGames AS ug

JOIN Users AS u ON u.Id = ug.UserId

JOIN Games AS g ON g.Id = ug.GameId

WHERE g.[Name] = @GameName AND

u.Username = @Username

--u.Username IN('baleremuda', 'loosenoise','inguinalself','buildingdeltoid','monoxidecos')

END

EXECUTE usp\_GiveBonusInCash 'Bali', 'baleremuda', 50000

EXECUTE usp\_GiveBonusInCash 'Bali', 'loosenoise', 50000

EXECUTE usp\_GiveBonusInCash 'Bali', 'inguinalself', 50000

EXECUTE usp\_GiveBonusInCash 'Bali', 'buildingdeltoid', 50000

EXECUTE usp\_GiveBonusInCash 'Bali', 'monoxidecos', 50000

--Another adding bonus cash as updating table, because

--according to exercise conditions (problem 19 - item 2) bonus is 50000,

--but in some solutions exist additions to this bonus

--(it must be execute only this query below or usp\_GiveBonusInCash)

UPDATE UsersGames

SET Cash += 50000 + (SELECT SUM(i.Price) FROM Items AS i JOIN UserGameItems AS ugi ON ugi.ItemId = i.Id

WHERE ugi.UserGameId = UsersGames.Id)

WHERE UserId IN (SELECT Id FROM Users

WHERE Username IN('baleremuda', 'loosenoise','inguinalself','buildingdeltoid','monoxidecos'))

AND GameId = (SELECT Id FROM Games WHERE [Name] = 'Bali')

DECLARE @bonusCash MONEY = 50000 +

(SELECT SUM(i.Price) FROM Items AS i JOIN UserGameItems AS ugi ON ugi.ItemId = i.Id

JOIN UsersGames AS ug ON ug.Id = ugi.UserGameId)

EXECUTE usp\_GiveBonusInCash 'Bali', 'baleremuda', @bonusCash

EXECUTE usp\_GiveBonusInCash 'Bali', 'loosenoise', @bonusCash

EXECUTE usp\_GiveBonusInCash 'Bali', 'inguinalself', @bonusCash

EXECUTE usp\_GiveBonusInCash 'Bali', 'buildingdeltoid', @bonusCash

EXECUTE usp\_GiveBonusInCash 'Bali', 'monoxidecos', @bonusCash

--Buying the items

GO

CREATE PROCEDURE usp\_BuyingItem(@GameName NVARCHAR(50), @Username NVARCHAR(50), @ItemId INT) AS

BEGIN

INSERT INTO UserGameItems

VALUES

(@ItemId, (SELECT ug.Id FROM UsersGames AS ug

JOIN Users AS u ON u.Id = ug.UserId

JOIN Games AS g ON g.Id = ug.GameId

WHERE g.[Name] = @GameName AND u.Username = @Username))

UPDATE ug SET ug.Cash -= (SELECT Price FROM Items AS i WHERE i.Id = @ItemId)

FROM UsersGames AS ug

JOIN Users AS u ON u.Id = ug.UserId

JOIN Games AS g ON g.Id = ug.GameId

WHERE g.[Name] = @GameName AND

u.Username = @Username

END

GO

CREATE PROCEDURE usp\_BuyingItems(@GameName NVARCHAR(50), @Username NVARCHAR(50),

@StartItemId INT, @EndItemId INT) AS

BEGIN

DECLARE @currentItemId INT = @StartItemId

WHILE(@currentItemId <= @EndItemId)

BEGIN

EXECUTE usp\_BuyingItem @GameName, @Username, @currentItemId

SET @currentItemId += 1

END

END

EXECUTE usp\_BuyingItems 'Bali', 'baleremuda', 251, 299

EXECUTE usp\_BuyingItems 'Bali', 'loosenoise', 251, 299

EXECUTE usp\_BuyingItems 'Bali', 'inguinalself', 251, 299

EXECUTE usp\_BuyingItems 'Bali', 'buildingdeltoid', 251, 299

EXECUTE usp\_BuyingItems 'Bali', 'monoxidecos', 251, 299

EXECUTE usp\_BuyingItems 'Bali', 'baleremuda', 501, 539

EXECUTE usp\_BuyingItems 'Bali', 'loosenoise', 501, 539

EXECUTE usp\_BuyingItems 'Bali', 'inguinalself', 501, 539

EXECUTE usp\_BuyingItems 'Bali', 'buildingdeltoid', 501, 539

EXECUTE usp\_BuyingItems 'Bali', 'monoxidecos', 501, 539

--Selecting users and thier items

SELECT u.Username, g.[Name], ug.Cash, i.[Name] AS [Item Name]

FROM Users AS u

JOIN UsersGames AS ug ON ug.UserId = u.Id

JOIN Games AS g ON g.Id = ug.GameId

JOIN UserGameItems AS ugi ON ugi.UserGameId = ug.Id

JOIN Items AS i ON i.Id = ugi.ItemId

WHERE g.[Name] = 'Bali'

ORDER BY u.Username, i.[Name]

--This below is another solution

--The Database Diablo must be dropt and create for receiving the correct result from exercises

GO

CREATE TRIGGER tr\_UserGameItems\_LevelRestriction ON UserGameItems

INSTEAD OF INSERT

AS

INSERT INTO UserGameItems

SELECT ItemId, UserGameId FROM inserted

WHERE ItemId IN

(SELECT Id FROM Items WHERE MinLevel <= (SELECT [Level] FROM UsersGames WHERE Id = UserGameId))

UPDATE UsersGames

SET Cash += 50000 + (SELECT SUM(i.Price) FROM Items AS i JOIN UserGameItems AS ugi ON ugi.ItemId = i.Id

WHERE ugi.UserGameId = UsersGames.Id)

WHERE UserId IN (SELECT Id FROM Users

WHERE Username IN('baleremuda', 'loosenoise','inguinalself','buildingdeltoid','monoxidecos'))

AND GameId = (SELECT Id FROM Games WHERE [Name] = 'Bali')

INSERT INTO UserGameItems(UserGameId, ItemId)

SELECT UsersGames.Id, i.Id FROM UsersGames, Items AS i

WHERE UserId IN (SELECT Id FROM Users

WHERE Username IN('baleremuda', 'loosenoise','inguinalself','buildingdeltoid','monoxidecos'))

AND GameId = (SELECT Id FROM Games WHERE [Name] = 'Bali')

AND ((i.Id > 250 AND i.Id < 300) OR (i.Id > 500 AND i.Id < 540))

UPDATE UsersGames

SET Cash -= (SELECT SUM(i.Price) FROM Items AS i JOIN UserGameItems AS ugi ON ugi.ItemId = i.Id

WHERE ugi.UserGameId = UsersGames.Id)

WHERE UserId IN (SELECT Id FROM Users

WHERE Username IN('baleremuda', 'loosenoise','inguinalself','buildingdeltoid','monoxidecos'))

AND GameId = (SELECT Id FROM Games WHERE [Name] = 'Bali')

SELECT u.Username, g.[Name], ug.Cash, i.[Name] AS [Item Name]

FROM Users AS u

JOIN UsersGames AS ug ON ug.UserId = u.Id

JOIN Games AS g ON g.Id = ug.GameId

JOIN UserGameItems AS ugi ON ugi.UserGameId = ug.Id

JOIN Items AS i ON i.Id = ugi.ItemId

WHERE g.[Name] = 'Bali'

ORDER BY u.Username, i.[Name]/\* 20MassiveShopping\*/

DECLARE @gameId INT, @sumForLevel11To12 MONEY, @sumForLevel19To21 MONEY

SELECT @gameId = ug.Id FROM UsersGames AS ug

JOIN Games AS g ON g.Id = ug.GameId WHERE g.[Name] = 'Safflower'

SET @sumForLevel11To12 = (SELECT SUM(i.Price) FROM Items AS i WHERE MinLevel BETWEEN 11 AND 12)

SET @sumForLevel19To21 = (SELECT SUM(i.Price) FROM Items AS i WHERE MinLevel BETWEEN 19 AND 21)

BEGIN TRANSACTION

IF ((SELECT Cash FROM UsersGames WHERE Id = @gameId) < @sumForLevel11To12)

ROLLBACK

ELSE

BEGIN

UPDATE UsersGames SET Cash -= @sumForLevel11To12 WHERE Id = @gameId

INSERT INTO UserGameItems(UserGameId, ItemId) SELECT @gameId, Id FROM Items

WHERE MinLevel BETWEEN 11 AND 12

COMMIT

END

BEGIN TRANSACTION

IF ((SELECT Cash FROM UsersGames WHERE Id = @gameId) < @sumForLevel19To21)

ROLLBACK

ELSE

BEGIN

UPDATE UsersGames SET Cash -= @sumForLevel19To21 WHERE Id = @gameId

INSERT INTO UserGameItems(UserGameId, ItemId)

SELECT @gameId, Id FROM Items WHERE MinLevel BETWEEN 19 AND 21

COMMIT

END

SELECT i.[Name] AS [Item Name]

FROM UserGameItems AS ugi

JOIN Items AS i ON i.Id = ugi.ItemId

WHERE ugi.UserGameId = @gameId

--This below doesn't work good - Judge doesen't accept it, but it must be more correctly than this above

BEGIN TRANSACTION

BEGIN TRY

INSERT INTO UserGameItems

SELECT j.Id, j.UserGameId FROM

(

SELECT i.Id, ugi.UserGameId FROM Users AS u

JOIN UsersGames AS ug ON ug.UserId = u.Id

JOIN Games AS g ON g.Id = ug.GameId

JOIN UserGameItems AS ugi ON ugi.UserGameId = ug.Id

JOIN Items AS i ON i.Id = ugi.ItemId

WHERE u.FirstName = 'Stamat' AND g.[Name] = 'Safflower' AND

i.Id NOT IN(SELECT ugi1.ItemId FROM UserGameItems AS ugi1 WHERE ugi1.UserGameId = ugi.UserGameId)

) AS j

END TRY

BEGIN CATCH

ROLLBACK

SELECT ERROR\_MESSAGE()

END CATCH

UPDATE ug SET ug.Cash -= i.Price

FROM Users AS u

JOIN UsersGames AS ug ON ug.UserId = u.Id

JOIN Games AS g ON g.Id = ug.GameId

JOIN UserGameItems AS ugi ON ugi.UserGameId = ug.Id

JOIN Items AS i ON i.Id = ugi.ItemId

WHERE u.FirstName = 'Stamat' AND g.[Name] = 'Safflower' AND

i.Id NOT IN(SELECT ugi1.ItemId FROM UserGameItems AS ugi1 WHERE ugi1.UserGameId = ugi.UserGameId)

IF ((SELECT ug.Cash FROM Users AS u

JOIN UsersGames AS ug ON ug.UserId = u.Id

JOIN Games AS g ON g.Id = ug.GameId

WHERE u.FirstName = 'Stamat' AND g.[Name] = 'Safflower') < 0)

BEGIN

ROLLBACK

RAISERROR('Cash is not enough!', 16, 1)

END

COMMIT

SELECT i.[Name] AS [Item Name]

FROM Items AS i

JOIN UserGameItems AS ugi ON ugi.ItemId = i.Id

JOIN UsersGames AS ug ON ug.Id = ugi.UserGameId

JOIN Games AS g ON g.Id = ug.GameId

JOIN Users AS u ON u.Id = ug.UserId

WHERE u.FirstName = 'Stamat' AND g.[Name] = 'Safflower'

ORDER BY i.[Name]

DROP DATABASE SoftUni

/\* 21EmployeesWithThreeProjects\*/

CREATE PROCEDURE usp\_AssignProject(@EmployeeID INT, @ProjectID INT) AS

BEGIN

DECLARE @maxEmployeeProgectsCount INT = 3

DECLARE @employeeProjectsCount INT

SET @employeeProjectsCount = (SELECT COUNT(\*) FROM EmployeesProjects WHERE EmployeeID = @EmployeeID)

BEGIN TRANSACTION

INSERT INTO EmployeesProjects (EmployeeID, ProjectID)

VALUES (@EmployeeID, @ProjectID)

IF (@employeeProjectsCount >= @maxEmployeeProgectsCount)

BEGIN

RAISERROR('The employee has too many projects!', 16, 1)

ROLLBACK

RETURN

END

COMMIT

END

/\* 22DeleteEmployees\*/

CREATE TABLE Deleted\_Employees

(

EmployeeId INT NOT NULL IDENTITY CONSTRAINT PK\_Deleted\_Employees PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

MiddleName VARCHAR(50),

JobTitle VARCHAR NOT NULL,

DepartmentId INT NOT NULL

CONSTRAINT FK\_Deleted\_Employees\_Departments FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary MONEY NOT NULL

)

--In Judge must be paste only the Trigger's creation below

GO

--The query below doesn't past in Judge - Compile time error

CREATE TRIGGER tr\_Deleted\_Employees\_Employees\_AfterFirring ON Employees

AFTER DELETE AS

BEGIN

--by inserting in this way it may take more as one value and this causes the error

INSERT INTO Deleted\_Employees

VALUES

(

(SELECT FirstName FROM deleted),

(SELECT LastName FROM deleted),

(SELECT MiddleName FROM deleted),

(SELECT JobTitle FROM deleted),

(SELECT DepartmentID FROM deleted),

(SELECT Salary FROM deleted)

)

END

--Only one query must be paste in Judge

GO

CREATE TRIGGER tr\_Deleted\_Employees\_Employees\_AfterFirring ON Employees

FOR DELETE AS

BEGIN

INSERT INTO Deleted\_Employees

SELECT FirstName, LastName, MiddleName, JobTitle, DepartmentID, Salary

FROM deleted

END

--Only one query must be paste in Judge

GO

CREATE TRIGGER tr\_Deleted\_Employees\_Employees\_AfterFirring ON Employees

AFTER DELETE AS

BEGIN

INSERT INTO Deleted\_Employees

SELECT FirstName, LastName, MiddleName, JobTitle, DepartmentID, Salary

FROM deleted

END

--In Judge must be paste without this below

ALTER TABLE Employees ALTER COLUMN ManagerID INT

ALTER TABLE Employees ALTER COLUMN DepartmentID INT

UPDATE Employees SET DepartmentID = NULL WHERE EmployeeID = 293

UPDATE Employees SET ManagerID = NULL WHERE EmployeeID = 293

DELETE FROM EmployeesProjects WHERE EmployeeID = 293

--this query below work only when the trigger isn't created

DELETE FROM Employees WHERE EmployeeID = 293

SELECT \* FROM Employees

--the table is empty because the delete query work only when the trigger isn't created

SELECT \* FROM Deleted\_Employees

--the commeted rows below doesn't work - drop Database SoftUni and create it again

--ALTER TABLE Employees ALTER COLUMN ManagerID INT NOT NULL

--ALTER TABLE Employees ALTER COLUMN DepartmentID INT NOT NULL

--INSERT INTO Employees (EmployeeID, FirstName, LastName, MiddleName, JobTitle, DepartmentID, ManagerID, HireDate, Salary, AddressID)

--VALUES (293, 'George', 'Denchev', NULL, 'Independent Java Consultant', 6, NULL, '20050301', 48000, 291)

--INSERT INTO Employees (EmployeeID, FirstName, LastName, MiddleName, JobTitle, DepartmentID, ManagerID, HireDate, Salary, AddressID)

--VALUES (292, 'Martin', 'Kulov', NULL, 'Independent .NET Consultant', 6, NULL, '20050301', 48000, 291)

--INSERT INTO Employees (EmployeeID, FirstName, LastName, MiddleName, JobTitle, DepartmentID, ManagerID, HireDate, Salary, AddressID)

--VALUES (291, 'Svetlin', 'Nakov', 'Ivanov', 'Independent Software Development Consultant', 6, NULL, '20050301', 48000, 291)

--INSERT INTO Employees (EmployeeID, FirstName, LastName, MiddleName, JobTitle, DepartmentID, ManagerID, HireDate, Salary, AddressID)

--VALUES (290, 'Lynn', 'Tsoflias', '', 'Sales Representative', 3, 288, '20050701', 23100, 153)

DROP TRIGGER tr\_Deleted\_Employees\_Employees\_AfterFirring