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 Bank
GO

CREATE PROC usp_SelectFullNamesOfPersons
AS
	SELECT FirstName + ' ' + LastName AS [Full Name]
	FROM Persons
GO

EXEC usp_SelectFullNamesOfPersons
```

2. ***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 Bank
GO

CREATE PROC usp_SelectPersonsByMoney(
	@amount money)
AS
	SELECT
		p.FirstName + ' ' + p.LastName AS [Full Name],
		a.Balance
	FROM Persons p
		INNER JOIN Accounts a
			ON p.PersonId = a.PersonId
	WHERE a.Balance >= @amount
GO

EXEC usp_SelectPersonsByMoney 10
```

3. ***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 Bank
GO

CREATE FUNCTION fn_caluculateInterestRate(@sum money, @yearlyInterestRate real,
@numberOfMonths int)
	RETURNS money
AS
BEGIN
	DECLARE @rate money = @yearlyInterestRate/12*@numberOfMonths
	RETURN @sum+@rate*@sum
END
GO

SELECT dbo.fn_caluculateInterestRate(10, 4, 6)
```

4. ***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 Bank
GO

CREATE PROC usp_interestRateToAccount(@accoundId int, @interestRate real)
AS
	SELECT
		Balance,
		dbo.fn_caluculateInterestRate(Balance, @interestRate, 1) AS [With
Interest]

	FROM Accounts
	WHERE AccountId = @accoundId
GO

EXEC usp_interestRateToAccount 2, 2.2
```

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

```
USE Bank
GO

CREATE PROC usp_withdrawMoney(@accoundId int, @amount money)
AS
	BEGIN TRAN
	UPDATE Accounts
	SET Balance = Balance - @amount
	WHERE AccountId = @accoundId
	COMMIT TRAN
GO

CREATE PROC usp_depositMoney(@accountId int, @amount money)
AS
	BEGIN TRAN
	UPDATE Accounts
	SET Balance = Balance + @amount
	WHERE AccountId = @accountId
	COMMIT TRAN
GO

EXEC usp_depositMoney 1, 200
EXEC usp_withdrawMoney 1, 199
```

6. ***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 Bank
GO
IF (OBJECT_ID('Logs') IS NULL)
BEGIN
	CREATE TABLE Logs(
		LogId int IDENTITY,
		AccountId int NOT NULL,
		OldSum money,
		NewSum money
```

```sql
				CONSTRAINT PK_LogId PRIMARY KEY(LogId)
				CONSTRAINT FK_Logs_Accounts
								FOREIGN KEY (AccountId)
								REFERENCES Accounts(AccountId)
		)
END
GO

USE Bank
GO

IF (OBJECT_ID('tr_OnAccountBalanceChange') IS NOT NULL)
	BEGIN
		DROP TRIGGER tr_OnAccountBalanceChange
	END
GO

CREATE TRIGGER tr_OnAccountBalanceChange
ON Accounts FOR UPDATE
AS
	DECLARE @accountId int, @oldSum money, @newSum money
	SELECT @accountId=d.AccountId, @oldSum=d.Balance
	FROM deleted d
	SELECT @newSum=i.Balance
	FROM inserted i
	INSERT INTO Logs(AccountId, OldSum, NewSum)
	VALUES (@accountId, @oldSum, @newSum)
GO

EXEC usp_depositMoney 1, 222
EXEC usp_withdrawMoney 1, 156
```

7. ***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'.***

```sql
USE TelerikAcademy
GO

CREATE FUNCTION fn_NameContainingLetters(
	@name nvarchar(50),
	@letters nvarchar(50)
	)
	RETURNS bit
AS
BEGIN
	DECLARE @contains bit
	SET @contains = 1
	DECLARE @currentLetter nvarchar(1)
	DECLARE @counter int = 1

	WHILE(@counter <= LEN(@name))
		BEGIN
			SET @currentLetter = SUBSTRING(@name, @counter, 1)
			IF(CHARINDEX(@currentLetter, @letters) = 0)
				BEGIN
					SET @contains = 0
					RETURN @contains
				END
			SET @counter = @counter + 1
		END
	RETURN @contains
END
GO

CREATE PROC usp_FindFirstName(
	@letters nvarchar(50)
	)
AS
	SELECT FirstName
	FROM Employees
	WHERE
	(SELECT dbo.fn_NameContainingLetters(FirstName, @letters)) = 1
GO

CREATE PROC usp_FindMiddleName(
	@letters nvarchar(50)
	)
AS
	SELECT MiddleName
	FROM Employees
	WHERE
	(SELECT dbo.fn_NameContainingLetters(MiddleName, @letters)) = 1
GO

CREATE PROC usp_FindLastName(
	@letters nvarchar(50)
	)
AS
	SELECT LastName
	FROM Employees
	WHERE
	(SELECT dbo.fn_NameContainingLetters(LastName, @letters)) = 1
GO

CREATE PROC usp_FindTown(
	@letters nvarchar(50)
	)
AS
	SELECT Name
	FROM Towns
	WHERE
	(SELECT dbo.fn_NameContainingLetters(Name, @letters)) = 1
GO

CREATE PROC usp_FindAllNames (@letters nvarchar(50))
AS
	EXEC dbo.usp_FindFirstName @letters
	--EXEC dbo.usp_FindMiddleName @letters
	EXEC dbo.usp_FindLastName @letters
	EXEC dbo.usp_FindTown @letters
GO

EXEC usp_FindAllNames 'oistmiahf'
```