# Session 2

FUNCTIONS, PROCEDURES

## Content

- Transact SQL (T-SQL)
- Functions
- Stored procedure

## 1. Transact SQL

- Proprietary extension of SQL
  - User-define functions, stored procedures, triggers, transactions,…
  - Delete: on joined tables
  - Bulk insert: insert from external data sources (text, csv,…)
- Procedural programming
  - Variables
  - Control flows (if..then, while, begin…end, goto,try…catch)

## 1. Transact SQL - Example

```
DECLARE @v int, @p varchar(10);
SET @v = 0;
DECLARE c CURSOR
   FOR SELECT employee_id FROM employees;
OPEN c;
FETCH NEXT FROM c INTO @p;
WHILE @@FETCH_STATUS = 0
BEGIN
   SET @v = @v + 1
   FETCH NEXT FROM c INTO @p;
END
IF @v < 100
PRINT 'Small';
ELSE
PRINT 'BIG';
CLOSE c;
DEALLOCATE c;
```

## 2. Functions

- System functions
  - Predefined functions
- User-defined function
  - Defined by programmers
  - Written in T-SQL
- Types of functions
  - Scalar functions: CONCAT, MONTH, YEAR,…
  - Table-valued functions: return a table
  - Aggregate functions: SUM, COUNT

## 2.1. Scalar function

```
CREATE FUNCTION schema_name.function_name (parameter_list)
RETURNS data_type AS
BEGIN
    statements
    RETURN value
END
```

## 2.1. Scalar function - example

```
CREATE FUNCTION experience(@hire_date DATE)
RETURNS INT AS
BEGIN
    RETURN YEAR(GETDATE()) - YEAR(@hire_date);
END;

…

SELECT dbo.experience(hire_date) FROM employees;
```

## 2.2. Table-valued function

```
CREATE FUNCTION schema_name.function_name (parameter_list)
RETURNS TABLE AS
    RETURN
        SELECT ….
```

## 2.2. Table-valued function

```
CREATE FUNCTION EMPLOYEES_BY_YEAR(@y int)
RETURNS TABLE AS
    RETURN
        SELECT * FROM employees
        WHERE YEAR(hire_date) = @y;


…
```

## 3. Stored procedure

- Group a sequence of T-SQL statements into a single execution unit
- Can be used to implement some business logic at database level

## 3. Stored procedure - Syntax

```
CREATE PROCEDURE schema_name.procedure_name (parameter_list)
AS
BEGIN
        statements;
END ;



EXECUTE schema_name.procedure_name param=value,….,param=value;
```

## 3. Stored procedure - Example

```
CREATE PROCEDURE UpdateSalary(@fn varchar(20), @ln varchar(20), @s real)
AS
BEGIN
    UPDATE employees SET salary = @s
        WHERE first_name = @fn AND last_name = @ln;
END


EXECUTE dbo.UpdateSalary @fn = 'a', @ln = 'b', @s = 10000;
```

## 4. Assignments

1. Create fullname function which concat the firstname and the lastname into a string value. Use this function for listing fullname of all employees.
2. Create get_job_title function which return a job title for a given job id. Use this function for listing all employees' names and their job title
3. Create a function that count the number of departments in a specific country. Using this function for counting number of departments in every regions
4. Create a procedure for changing job of an employee:
   - Parameters: employee_id, job_id
   - Process:
     - Add current job of the employee as a job history
     - Update job_id, salary ( as the min salary of the job) for the employee record