Functions & Procedures – Exercises

office@s

1. Database Setup

```
CREATE DATABASE PracticeDB;
GO
USE PracticeDB;
GO

CREATE TABLE Employees (
    EmployeeId INT PRIMARY KEY IDENTITY(1,1),
    Name NVARCHAR(50),
    DateOfBirth DATE,
    DepartmentId INT,
    Salary DECIMAL(10,2)
);

CREATE TABLE Departments (
    DepartmentId INT PRIMARY KEY IDENTITY(1,1),
    DepartmentName NVARCHAR(50)
);

CREATE TABLE Products (
    ProductId INT PRIMARY KEY IDENTITY(1,1),
    ProductName NVARCHAR(50),
    CategoryId INT,
    Price DECIMAL(10,2),
```

.

```sql
    Stock INT
);

CREATE TABLE Categories (
    CategoryId INT PRIMARY KEY IDENTITY(1,1),
    CategoryName NVARCHAR(50)
);

--Insert data
INSERT INTO Departments (DepartmentName) VALUES ('HR'), ('IT'),
('Sales'), ('Marketing');

INSERT INTO Employees (Name, DateOfBirth, DepartmentId, Salary)
VALUES
('John Doe', '1990-06-15', 1, 50000),
('Jane Smith', '1985-12-22', 2, 70000),
('Alice Brown', '1992-03-10', 3, 45000),
('Bob Johnson', '1988-09-05', 4, 55000);

INSERT INTO Categories (CategoryName) VALUES ('Electronics'),
('Clothing'), ('Home Appliances');

INSERT INTO Products (ProductName, CategoryId, Price, Stock)
VALUES
('Smartphone', 1, 699.99, 50),
('Laptop', 1, 1299.99, 30),
('T-Shirt', 2, 19.99, 100),

('Vacuum Cleaner', 3, 149.99, 20);
```

2. Scalar Functions

Create a scalar function GetFullYear that extracts the year from a given date. Test it using GETDATE().

`SELECT dbo.GetFullYear(GETDATE());`	2025

Write a scalar function GetAnnualSalary that calculates the annual salary of an employee based on their monthly salary.

.

`SELECT dbo.GetAnnualSalary(5000);`	60000

Create a function IsInStock that takes a product ID and returns TRUE if the stock is greater than 0, otherwise FALSE.

`SELECT dbo.IsInStock(1); -- Smartphone`	TRUE

Develop a scalar function GetDiscountPrice that takes a price and a discount percentage and returns the discounted price.

`SELECT dbo.GetDiscountPrice(699.99, 10);`	629.99

3. Inline Table-Valued Functions

Create an inline TVF GetEmployeesByDepartment that returns all employees belonging to a given department.

`SELECT * FROM dbo.GetEmployeesByDepartment(2);`	EmployeeId | Name | DepartmentId | Salary --- 2 | Jane Smith | 2 | 70000

Write an inline TVF GetProductsByCategory that returns all products for a given category ID.

`SELECT * FROM dbo.GetProductsByCategory(1);`	ProductId | ProductName | CategoryId | Price | Stock 1 | Smartphone | 1 | 699.99 | 50 2 | Laptop | 1 | 1299.99 | 30

Create a function GetAffordableProducts that takes a maximum price as input and returns all products below that price.

.

`SELECT * FROM dbo.GetAffordableProducts(100);`	3 | T-Shirt | 19.99

Write a function GetDepartmentsWithEmployees that returns all departments with at least one employee.

`SELECT * FROM dbo.GetDepartmentsWithEmployees();`	1 | HR 2 | IT 3 | Sales 4 | Marketing

4. Multi-Statement Table-Valued Functions

Create a multi-statement TVF GetTopPaidEmployees that takes a department ID and returns the top 3 employees with the highest salaries in that department.

`SELECT * FROM dbo.GetTopPaidEmployees(3);`	3 | Alice Brown| 45000

Write a function GetEmployeeDetails that returns a table with employee names, their department names, and salaries.

`SELECT * FROM dbo.GetEmployeeDetails();`	1 | John Doe | HR | 50000 2 | Jane Smith | IT | 70000

Develop a multi-statement TVF GetOutOfStockProducts that returns all products where the stock is 0.

`SELECT * FROM dbo.GetOutOfStockProducts();`	

Create a TVF GetEmployeesByAgeRange that takes a minimum and maximum age and returns employees whose age falls within the range.

`SELECT * FROM dbo.GetEmployeesByAgeRange(30, 40);`	1 | John Doe | 35 2 | Jane Smith | 39

.

5. Stored Procedures

Create a stored procedure UpdateSalary that takes an employee ID and a percentage and increases their salary by that percentage.

`EXEC UpdateSalary 1, 10;`	1 | 55000

Write a stored procedure AddNewProduct to insert a new product into the Products table.

`EXEC AddNewProduct 'Smartwatch', 1, 199.99, 100;`	Product added successfully.

office@

Develop a stored procedure DeleteLowStockProducts that deletes all products with stock below 5.

`EXEC DeleteLowStockProducts;`	Products with low stock deleted.

Write a procedure TransferEmployee that takes an employee ID and a department ID and moves the employee to the new department.

`EXEC TransferEmployee 4, 2;`	4 | Bob Johnson| 2

6. Optional Advanced Task

Create a multi-result stored procedure GetEmployeeAndDepartmentInfo that returns:

- A result set of all employees.
- A result set of all departments.

.