-- Create Database

CREATE DATABASE CompanyDB

USE CompanyDB;

-- Create Department Table

CREATE TABLE Department (

DeptID INT PRIMARY KEY,

DeptName VARCHAR(50)

);

-- Create Employee Table

CREATE TABLE Employee (

EmpID INT PRIMARY KEY,

EmpName VARCHAR(50),

Salary DECIMAL(10,2),

DeptID INT FOREIGN KEY REFERENCES Department(DeptID)

);

-- Insert data into Department table

INSERT INTO Department VALUES (1, 'HR'), (2, 'IT'), (3, 'Finance');

-- Insert data into Employee table

INSERT INTO Employee VALUES

(101, 'Ross', 50000, 1),

(102, 'Monica', 60000, 2),

(103, 'Chandler', 55000, 2),

(104, 'Rachel', 70000, 3),

(105, 'Joey', 45000, 1),

(106, 'Phoeboe', 75000, 1);

select \* from Department

select \* from Employee

-- Stored Procedure to update salary if certain condition is met

CREATE PROCEDURE UpdateSalary

@EmpID INT,

@Increment DECIMAL(10,2)

AS

BEGIN

-- Declare variables to hold the output values

DECLARE @CurrentSalary DECIMAL(10,2);

-- to update the salary

SELECT @CurrentSalary = Salary FROM Employee WHERE EmpID = @EmpID;

IF @CurrentSalary < 60000

UPDATE Employee SET Salary = Salary + @Increment WHERE EmpID = @EmpID;

ELSE

PRINT 'Salary already high, no increment applied';

END;

-- Call the stored procedure

EXEC UpdateSalary @EmpID = 105, @Increment = 5000;

-- To see the output

SELECT EmpID, EmpName, Salary FROM Employee WHERE EmpID = 105;

-- Function to calculate annual salary

CREATE FUNCTION fn\_AnnualSalary (@EmpID INT)

RETURNS DECIMAL(10,2)

AS

BEGIN

DECLARE @AnnualSalary DECIMAL(10,2);

SELECT @AnnualSalary = Salary \* 12 FROM Employee WHERE EmpID = @EmpID;

RETURN @AnnualSalary;

END;

-- Call the function

SELECT EmpName, dbo.fn\_AnnualSalary(EmpID) AS AnnualSalary FROM Employee;

--IN & OUT parameters

CREATE OR ALTER PROCEDURE Sampleroc

@Input INT, -- IN parameter

@Output INT OUTPUT -- OUT parameter

AS

BEGIN

SET @Output = @Input \* 2;

END;

GO

DECLARE @Result INT;

EXEC Sampleroc @Input = 10, @Output = @Result OUTPUT;

-- Display the result in the Results tab

SELECT @Result AS OutputValue;

-- Create a table-valued function

CREATE FUNCTION GetEmployeesByDept (@departmentName VARCHAR(50))

RETURNS TABLE

AS

RETURN (

SELECT e.EmpID, e.EmpName, e.Salary, d.DeptName

FROM Employee e

JOIN Department d ON e.DeptID = d.DeptID

WHERE d.DeptName = @departmentName

);

-- Call the table-valued function

SELECT \* FROM dbo.GetEmployeesByDept('HR');

--Loops

CREATE PROCEDURE PrintNumbers

AS

BEGIN

DECLARE @i INT = 1;

WHILE @i <= 5

BEGIN

PRINT @i;

SET @i = @i + 1;

END

END;

EXEC PrintNumbers

-- Example: Trigger after insert on Employee

CREATE TRIGGER trg\_AfterInsert

ON Employee

AFTER INSERT

AS

BEGIN

PRINT 'New Employee Added!';

END;

BEGIN TRY

EXEC UpdateSalary 999, 1000; -- Non-existing employee

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH;