**Exercise 1: Create a Scalar Function to Calculate Square of a Number**

**Task:**

Create a scalar function named fn\_Square that takes an integer as input and returns its square.

**Solution:**

CREATE FUNCTION fn\_Square (@Number INT)

RETURNS INT

AS

BEGIN

RETURN @Number \* @Number

END;

**Test:**

SELECT dbo.fn\_Square(5) AS SquareResult;

**Exercise 2: Create a Scalar Function to Calculate Factorial**

**Task:**

Create a scalar function fn\_Factorial that takes an integer as input and returns its factorial.

**Solution:**

CREATE FUNCTION fn\_Factorial (@Number INT)

RETURNS BIGINT

AS

BEGIN

DECLARE @Result BIGINT = 1;

DECLARE @Counter INT = 1;

WHILE @Counter <= @Number

BEGIN

SET @Result = @Result \* @Counter;

SET @Counter = @Counter + 1;

END;

RETURN @Result;

END;

**Test:**

SELECT dbo.fn\_Factorial(5) AS FactorialResult;

**Exercise 3: Create a Scalar Function to Convert Temperature from Celsius to Fahrenheit**

**Task:**

Create a scalar function fn\_CelsiusToFahrenheit that takes a float (Celsius) and converts it to Fahrenheit.

**Solution:**

CREATE FUNCTION fn\_CelsiusToFahrenheit (@Celsius FLOAT)

RETURNS FLOAT

AS

BEGIN

RETURN (@Celsius \* 9.0 / 5.0) + 32;

END;

**Test:**

SELECT dbo.fn\_CelsiusToFahrenheit(100) AS Fahrenheit;

**Exercise 4: Create an Inline Table-Valued Function to Get Employees by Department**

**Task:**

Create an inline table-valued function fn\_GetEmployeesByDept that takes a department name as input and returns all employees in that department.

**Solution:**

CREATE FUNCTION fn\_GetEmployeesByDept (@DeptName NVARCHAR(50))

RETURNS TABLE

AS

RETURN

(

SELECT EmployeeID, EmployeeName, Department, Salary

FROM Employees

WHERE Department = @DeptName

);

**Test:**

SELECT \* FROM dbo.fn\_GetEmployeesByDept('IT');

**Exercise 5: Create an Inline Table-Valued Function to Get Products by Category**

**Task:**

Create an inline table-valued function fn\_GetProductsByCategory that returns products of a given category.

**Solution:**

CREATE FUNCTION fn\_GetProductsByCategory (@CategoryID INT)

RETURNS TABLE

AS

RETURN

(

SELECT ProductID, ProductName, Price

FROM Products

WHERE CategoryID = @CategoryID

);

**Test:**

SELECT \* FROM dbo.fn\_GetProductsByCategory(2);

**Exercise 6: Create a Multi-Statement Table-Valued Function to Return Prime Numbers up to N**

**Task:**

Create a function fn\_GetPrimes that returns a table of prime numbers up to a given number.

**Solution:**

CREATE FUNCTION fn\_GetPrimes (@Max INT)

RETURNS @PrimeTable TABLE (Number INT)

AS

BEGIN

DECLARE @Counter INT = 2;

WHILE @Counter <= @Max

BEGIN

IF NOT EXISTS (SELECT 1 FROM @PrimeTable WHERE @Counter % Number = 0)

BEGIN

INSERT INTO @PrimeTable VALUES (@Counter);

END;

SET @Counter = @Counter + 1;

END;

RETURN;

END;

**Test:**

SELECT \* FROM dbo.fn\_GetPrimes(50);

**Exercise 7: Create a Function to Calculate Discounted Price**

**Task:**

Create a scalar function fn\_GetDiscountedPrice that takes a price and a discount percentage and returns the discounted price.

**Solution:**

CREATE FUNCTION fn\_GetDiscountedPrice (@Price DECIMAL(10,2), @Discount DECIMAL(5,2))

RETURNS DECIMAL(10,2)

AS

BEGIN

RETURN @Price - (@Price \* @Discount / 100);

END;

**Test:**

SELECT dbo.fn\_GetDiscountedPrice(1000, 10) AS DiscountedPrice;

**Exercise 8: Create a Function to Get Employee Salary Range**

**Task:**

Create an inline table-valued function fn\_GetSalaryRange that returns employees whose salary falls between a given range.

**Solution:**

CREATE FUNCTION fn\_GetSalaryRange (@MinSalary DECIMAL(10,2), @MaxSalary DECIMAL(10,2))

RETURNS TABLE

AS

RETURN

(

SELECT EmployeeID, EmployeeName, Salary

FROM Employees

WHERE Salary BETWEEN @MinSalary AND @MaxSalary

);

**Test:**

SELECT \* FROM dbo.fn\_GetSalaryRange(3000, 7000);

**Exercise 9: Create a Function to Get Customer Order Count**

**Task:**

Create a scalar function fn\_GetOrderCount that takes a customer ID and returns the total number of orders they have placed.

**Solution:**

CREATE FUNCTION fn\_GetOrderCount (@CustomerID INT)

RETURNS INT

AS

BEGIN

DECLARE @OrderCount INT;

SELECT @OrderCount = COUNT(\*) FROM Orders WHERE CustomerID = @CustomerID;

RETURN @OrderCount;

END;

**Test:**

SELECT dbo.fn\_GetOrderCount(5) AS OrderCount;

**Exercise 10: Create a Function to Get Student Grades Based on Marks**

**Task:**

Create a scalar function fn\_GetGrade that takes a student's marks and returns a grade.

**Solution:**

CREATE FUNCTION fn\_GetGrade (@Marks INT)

RETURNS CHAR(1)

AS

BEGIN

DECLARE @Grade CHAR(1);

IF @Marks >= 90 SET @Grade = 'A';

ELSE IF @Marks >= 80 SET @Grade = 'B';

ELSE IF @Marks >= 70 SET @Grade = 'C';

ELSE IF @Marks >= 60 SET @Grade = 'D';

ELSE SET @Grade = 'F';

RETURN @Grade;

END;

**Test:**

SELECT dbo.fn\_GetGrade(85) AS Grade;