From the above given tables perform the following queries:

Part – A

Scalar Valued Functions:

1. Create a function which displays total number of employees.

```
CREATE FUNCTION GetTotalEmployee( )
RETURNS INT
AS
BEGIN
        RETURN (SELECT COUNT(EID) FROM Employee)
END

SELECT dbo.GetTotalEmployee() as TotalEmployee;
```

2. Create a function which returns highest salary from Employee table.

```
CREATE FUNCTION GetEmployeeHigestSalary( )
RETURNS INT
AS
BEGIN
        RETURN (SELECT MAX(Salary) FROM Employee)
END

SELECT dbo.GetEmployeeHigestSalary() AS HigestSalary
```

3. Create a function to get the experience of the employee based on their joining date.

```
CREATE FUNCTION GetEmployeeExperience (@EmpID INT)
RETURNS INT
AS
BEGIN
   DECLARE @Experience INT;
        SELECT @Experience = DATEDIFF(YEAR, JoiningDate, GETDATE())
        FROM Employee
        WHERE EID = @EmpID;
   RETURN @Experience;
END;

SELECT dbo.GetEmployeeExperience(1)
```

4. Create a function that calculates the factorial of a given number.

```
CREATE FUNCTION GetFactorial(@n  INT)
RETURNS INT
AS
BEGIN
   DECLARE @FACT INT;
        SET @FACT = 1
        DECLARE @i  INT
```

```
        SET @i = 1
        WHILE @i <= @n
        BEGIN
            SET @FACT = @FACT * @i
            set @i = @i +1
        END
        RETURN @FACT
END;

SELECT dbo.GetFactorial(5)
```

5. Create a function which returns minimum salary of female employee.

```
CREATE FUNCTION GetMinimumSalaryFemale()
RETURNS INT
AS
BEGIN
   DECLARE @MinSalary int
        SELECT @MinSalary = Min(Salary)
        FROM Employee
        WHERE Gender = 'Female';
   RETURN @MinSalary
END

SELECT dbo.GetMinimumSalaryFemale() AS MinimumSalaryFemale
```

6. Create a function which count unique city from employee table.

```
ALTER FUNCTION CountUniqueCity()
RETURNS INT
AS
BEGIN
        RETURN (SELECT COUNT(DISTINCT CITY ) FROM Employee)
END

SELECT dbo.CountUniqueCity()
```

7. Create a Scalar-valued function that returns the name combined with salary of an employee based on their employee id and displayed output like ‘Roy having 3500 salaries’.

```
CREATE FUNCTION dbo.CombineNameWithSalary(@EID INT)
RETURNS VARCHAR(200)
AS
BEGIN
   DECLARE @Result VARCHAR(200);
        -- Fetch the employee's name and salary and format the result
        SELECT @Result = CONCAT(EName, ' having ', Salary, ' salaries')
        FROM Employee
```

```
        WHERE EID = @EID;
    RETURN @Result;
END;

SELECT dbo.CombineNameWithSalary(2)
```

Table Valued Functions:

1. Create a function which retrieve the data of Employee table.

```
CREATE FUNCTION GetAllDataFromEmployee()
RETURNS TABLE
AS
        RETURN (SELECT * FROM Employee)

SELECT * from dbo.GetAllDataFromEmployee()
```

2. Create a function which returns an employee table with city wise total salary.

```
CREATE FUNCTION GetTotalSalaryEmployee()
RETURNS TABLE
AS
        RETURN (SELECT City,SUM(Salary) AS TotalSalary
                FROM Employee
                GROUP BY City)

SELECT * FROM dbo.GetTotalSalaryEmployee()
```

3. Create a function which returns an employee table with gender wise maximum, minimum, total and average salaries.

```
CREATE FUNCTION GenderWiseMaxMinSalary()
RETURNS TABLE
AS
        RETURN (SELECT Gender,MAX(Salary) AS MaxSalary,
                        MIN(Salary) AS MinSalary ,
                        SUM(Salary) AS TotalSalary,
                        AVG(Salary) as AverageSalary
                FROM Employee
                GROUP BY Gender)

SELECT * FROM GenderWiseMaxMinSalary()
```

4. Create a function which return an employee table with details of employee whose name starts with J.

```
CREATE FUNCTION GetEmployee()
RETURNS TABLE
AS
        RETURN (SELECT * FROM Employee WHERE EName LIKE 'J%')
```

```
SELECT * FROM GetEmployee()
```

5. Create a function to get all the male employees.

```
CREATE FUNCTION GetAllEmployeeMale()
RETURNS TABLE
AS
        RETURN (SELECT * FROM Employee WHERE Gender = 'Male')

SELECT * FROM GetAllEmployeeMale()
```

6. Create a function to get employees from a given city.

```
CREATE FUNCTION GetEmployeeGivenCity(@City VARCHAR(100))
RETURNS TABLE
AS
        RETURN (SELECT * FROM Employee WHERE City = @City )

SELECT * FROM GetEmployeeGivenCity('London')
```

7. Create a function that displays employees with a salary greater than a specified amount.

```
CREATE FUNCTION GetEmployeeWithGretherAmount(@Amount int)
RETURNS TABLE
AS
        RETURN (SELECT * FROM Employee WHERE Salary > @Amount)

SELECT * FROM GetEmployeeWithGretherAmount(8000)
```

8. Create a function to get employees who joined after a given specified date.

```
CREATE FUNCTION GetEmployeeWithAfterDate(@GivenDate DATE)
RETURNS TABLE
AS
        RETURN (SELECT * FROM Employee WHERE JoiningDate > @GivenDate)

SELECT * FROM GetEmployeeWithAfterDate('2015-1-1')
```

PART B:

1. Create UDF to get the full name and department of an employee.

```
CREATE FUNCTION getFullNameDepartmentOfEmployee(@EID INT)
RETURNS VARCHAR(200)
AS
BEGIN
        RETURN (SELECT CONCAT (First_Name,' ',Last_Name,' ',Department)
                FROM Employee
                WHERE Employee_ID = @EID)
END

SELECT dbo.getFullNameDepartmentOfEmployee(1)
```

2. Create UDF to calculate the age of an employee based on the birth year.

```
CREATE FUNCTION dbo.CalculateEmployeeAge(@BirthYear INT)
RETURNS INT
AS
BEGIN
   DECLARE @Age INT;
        SET @Age = YEAR(GETDATE())-@BirthYear
        RETURN @Age
END

SELECT dbo.CalculateEmployeeAge(2013) as CurrentAge
```

3. Create UDF to get the number of employees in a specific department.

```
CREATE FUNCTION NumberEmployeeinDepartment(@Department varchar(100))
RETURNS INT
AS
BEGIN
        RETURN (SELECT COUNT(Employee_ID)
                FROM Employee
                WHERE Department = @Department)
END

SELECT dbo.NumberEmployeeinDepartment('HR')
```

4. Create UDF to concatenate the first name and last name with a custom separator.

```
CREATE FUNCTION CncatFirstnameLastname
        (       @FirstName NVARCHAR(100),
                @LastName NVARCHAR(100),
                @Separator NVARCHAR(10)
        )
RETURNS VARCHAR(200)
AS
BEGIN
        RETURN CONCAT(@FirstName, @Separator, @LastName);
END

SELECT dbo.CncatFirstnameLastname('John', 'Doe', ' - ') AS FullName;
```

5. Create UDF to check if an employee is part of the IT department.

```
CREATE FUNCTION dbo.IsEmployeeInIT(@EmployeeID INT)
RETURNS BIT
AS
BEGIN
        DECLARE @ISIT  BIT
        SELECT @ISIT = CASE
                        WHEN Department = 'IT' THEN 1
```

```
                ELSE 0
            END
        FROM Employee
        WHERE Employee_ID = @EmployeeID
        RETURN @ISIT
END

SELECT dbo.IsEmployeeInIT(3) AS IsInIT;
```

6. Create UDF to convert age into a friendly message.

```
CREATE FUNCTION GetFriendlyAgeMessage(@Age INT)
RETURNS VARCHAR(50)
AS
BEGIN
  DECLARE @Message NVARCHAR(50);
      SET @Message = CONCAT('YOU ARE ',@Age ,' YEARS OLD ')
  RETURN @Message;
END;

SELECT dbo.GetFriendlyAgeMessage(20)
```

7. Create UDF to find the average age of employees in a department.

```
CREATE FUNCTION GetAverageAgeByDepartment(@Department VARCHAR(50))
RETURNS FLOAT
AS
BEGIN
  DECLARE @AverageAge FLOAT;
      SELECT @AverageAge = AVG(CONVERT(float,Age))
      FROM Employee
      WHERE Department = @Department;
  RETURN @AverageAge;
END;

SELECT dbo.GetAverageAgeByDepartment('HR')
```

8. Create UDF to check if an employee exists in the table.

```
CREATE FUNCTION dbo.IsEmployeeExists(@EmployeeID INT)
RETURNS BIT
AS
BEGIN
  DECLARE @Exists BIT;
      SELECT @Exists = CASE
                WHEN COUNT(*) > 0 THEN 1
                ELSE 0
            END
      FROM Employee
```

```
        WHERE Employee_ID = @EmployeeID;
    RETURN @Exists;
END;

SELECT dbo.IsEmployeeExists(1) as EmployeeExists
```

9. Create UDF to get the last name in uppercase.

```
CREATE FUNCTION LastNameUpperCase(@LastName varchar(100))
RETURNS VARCHAR(200)
AS
BEGIN
        RETURN (SELECT UPPER(@LastName)
                FROM Employee
                WHERE Last_Name = @LastName)
END;

SELECT dbo.LastNameUpperCase('Doe')
```

PART C:

10. Create UDF to check if an employee is older than a specific age.

```
CREATE FUNCTION IsEmployeeOlderThan (@EmployeeID INT, @SpecificAge INT)
RETURNS BIT
AS
BEGIN
    DECLARE @IsOlder BIT;
        SELECT @IsOlder = CASE
                        WHEN Age > @SpecificAge THEN 1
                        ELSE 0
                                END
        FROM Employee
        WHERE Employee_ID = @EmployeeID
        RETURN @IsOlder
END;

SELECT dbo.IsEmployeeOlderThan(1,20)
```

11. Create UDF to get the first initial of an employee's first name.

```
CREATE FUNCTION GetFirstInitial(@EmployeeID INT)
RETURNS CHAR(1)
AS
BEGIN
    DECLARE @FirstInitial CHAR(1);
        SELECT @FirstInitial = LEFT(First_Name,1)
        FROM Employee
        WHERE Employee_ID = @EmployeeID
    RETURN @FirstInitial;
```

```
END;

SELECT dbo.GetFirstInitial(2) AS FirstInitial
```

12. Create UDF to get the number of employees older than a specific age.

```
CREATE FUNCTION GetEmployeesOlderThan(@SpecificAge INT)
RETURNS INT
AS
BEGIN
   DECLARE @EmployeeCount INT;
        SELECT @EmployeeCount = COUNT(*)
        FROM Employee
        WHERE Age > @SpecificAge;
   RETURN @EmployeeCount;
END;

 SELECT dbo.GetEmployeesOlderThan(20)
```

13. Create UDF to check if an employee's first name starts with a specific letter.

```
CREATE FUNCTION FirstNameStartWithLater(@EmployeeID INT,@Letter char(1))
RETURNS BIT
AS
BEGIN
        DECLARE @StartsWith BIT;
        SELECT @StartsWith = CASE
                                WHEN LEFT(First_Name,1) = @Letter THEN 1
                                ELSE 0
                             END
        FROM Employee
        WHERE Employee_ID = @EmployeeID
        RETURN @StartsWith;
END;

SELECT dbo.FirstNameStartWithLater(1,'j')
```

14. Create UDF to calculate the years of experience based on the current year and an employee's starting year.

```
CREATE FUNCTION CalculateExperience(@StartYear int)
RETURNS INT
AS
BEGIN
        DECLARE @YearsOfExperience INT;
        SET @YearsOfExperience = YEAR(GETDATE())- @StartYear
        RETURN @YearsOfExperience
END;

SELECT dbo.CalculateExperience(2023) AS StartYear
```