16. Functions

Refer below tables for functions.

Table: Appointments

Id	Date	DoctorId	PatientId
1	2020-06-12	1	2
2	2020-06-13	3	2
3	2020-06-14	3	1
4	2020-06-13	1	4
5	2020-06-13	3	6
6	2020-06-14	2	3
7	2020-06-15	2	2
8	2020-06-15	2	2

Table: Doctors

Id	Name
1	Williams
2	Smith
3	Clark
4	Johnson

Table: Patients

Id	Name
1	Robert
2	James
3	David
4	Michael
5	Oliver
6	Mary

Data Script:

```
CREATE TABLE Doctors(
    Id int PRIMARY KEY,
    Name varchar(50) NOT NULL
);

CREATE TABLE Patients(
    Id int PRIMARY KEY,
    Name varchar(50) NOT NULL
);

CREATE TABLE Appointments(
    Id int PRIMARY KEY,
    Date date NOT NULL,
    DoctorId int NOT NULL,
    PatientId int NOT NULL,
    FOREIGN KEY (PatientId) REFERENCES Patients(Id),
    FOREIGN KEY (DoctorId) REFERENCES Doctors(Id),
);

INSERT INTO Doctors(Id, Name)
VALUES (1, 'Williams'), (2, 'Smith'), (3, 'Clark'), (4, 'Johnson');

INSERT INTO Patients(Id, Name)
VALUES (1, 'Robert'), (2, 'James'), (3, 'David'), (4, 'Michael'), (5, 'Oliver'),
(6, 'Mary');

INSERT INTO Appointments(Id, Date, DoctorId, PatientId)
VALUES (1, '06/12/2020', 1, 2), (2, '06/13/2020', 3, 2),(3, '06/14/2020', 3, 1),
(4, '06/13/2020', 1, 4), (5, '06/13/2020', 3, 6), (6, '06/14/2020', 2, 3);
```

Functions

We can create two types of functions in SQL Server.

- Scalar-valued Functions
- Table-valued Functions

Scalar-valued Functions

Scalar valued functions return a value.

```
CREATE FUNCTION [dbo].[fnGetAppointmentDate] (@AppointmentId INT) RETURNS DATE

AS

BEGIN

DECLARE @Date DATE = (SELECT Date FROM Appointments WHERE Id = @AppointmentId);

      RETURN @Date;
END
```

Execute:

```
SELECT dbo.fnGetAppointmentDate(1);
```

Table-valued Functions

Table valued functions return a table.

```
CREATE FUNCTION [dbo].[fnGetAppointments] () RETURNS TABLE

AS
RETURN
(
      SELECT * FROM Appointments
);
Go
```

Execute:

```
SELECT * FROM fnGetAppointments();
```

Stored Procedures vs Functions

Stored Procedure	Functions
Output is not compulsory	Should have a output
Cannot be called within a function	Can be called within a stored procedure
Both input & output parameters are allowed	Only input parameters are allowed

Call Functions within a Function

Function 1:

```
CREATE FUNCTION [dbo].[FnGetPatientNameById] (@PatientId INT) RETURNS VARCHAR(50)

AS
BEGIN
	RETURN (SELECT Name FROM Patients WHERE Id = @PatientId)
END
```

Execute:

```
SELECT dbo.FnGetPatientNameById(1);
```

Function 2:

```
CREATE FUNCTION [dbo].[FnGetDoctorNameById] (@DoctorId INT) RETURNS VARCHAR(50)

AS
BEGIN
	RETURN (SELECT Name FROM Doctors WHERE Id = @DoctorId)
END
```

Execute:

```
SELECT dbo.FnGetDoctorNameById(1);
```

Function 3:

```
CREATE FUNCTION [dbo].[FnGetPatientNameAndDoctorName] (@PatientId INT, @DoctorId
INT) RETURNS VARCHAR(100)

AS
BEGIN
      RETURN (SELECT dbo.FnGetPatientNameById(@PatientId) + ' / ' +
dbo.FnGetDoctorNameById(@DoctorId))
END
```

Execute:

```
SELECT [dbo].[FnGetPatientNameAndDoctorName](1, 2) AS [Patient / Doctor]
```

Call Functions within a Stored Procedure

Stored Procedure:

```
CREATE PROCEDURE [dbo].[GetPatientNameAndDoctorName]
@PatientId INT, @DoctorId INT

AS
BEGIN
      SELECT dbo.FnGetPatientNameById(@PatientId) + ' / ' +
dbo.FnGetDoctorNameById(@DoctorId) AS [Patient / Doctor]
END
```

Execute:

```
EXEC GetPatientNameAndDoctorName 1, 2 ;
```

Calling Functions within a Stored Procedure

The below stored procedure is used to concatenate the patient's name and doctor's name.

Stored Procedure:

```
CREATE PROCEDURE [dbo].[GetPatientNameAndDoctorName]
@PatientId INT, @DoctorId INT
AS
BEGIN
      SELECT dbo.FnGetPatientNameById(@PatientId) + ' / ' +
dbo.FnGetDoctorNameById(@DoctorId) AS [Patient / Doctor]
END
```

Execute:

```
EXEC GetPatientNameAndDoctorName 1, 2;
```

Add two Numbers

```
CREATE FUNCTION FnAddTwoNumbers(@Number1 INT, @Number2 INT) RETURNS INT

AS
BEGIN
      RETURN Number1 + Number2
END
```