2022-2 데이터베이스시스템 Lab03

학번	2020136129	이름	최수연

Task -1: Write scalar functions

1. A function that sums the prime numbers and non-prime numbers between (1 and a given number say (@rng)) and returns their difference

```
IF OBJECT_ID('dbo.DiffNum') IS NOT NULL DROP FUNCTION dbo.DiffNum;
GO
CREATE FUNCTION dbo.DiffNum (@rng AS INT)
RETURNS INT
AS
BEGIN
DECLARE @primSum INT = 2, @ nonPrimSum INT = 1, @I INT = 3;
WHILE (@I <= @rng)
BEGIN
DECLARE @J INT = 2, @b BIT = 'true';
WHILE (@J * @J <= @rng)
BEGIN
IF (@I % @J = 0)
BEGIN
@nonPrimSum += @I
@b = 'false'
break
END
SET @J = @J + 1
END
IF (@b = 'true')
BEGIN
@primSum += @I
END
SET @I = @I + 1
END
RETURN (ABS(@primSum - @nonPrimSum))
END;
GO
```

2. (Pythagorean Triples) A right triangle can have sides whose lengths are all integers. The set of three integer values for the lengths of the sides of a right triangle is called a Pythagorean triple. The lengths of the three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse.
Write a table valued function that stores the Pythagorean triples for side1, side2 and the hypotenuse in a table variable, all between given numbers say (@srange, @erange), then returns the table

```
IF OBJECT_ID('dbo.PythagoreanTriples') IS NOT NULL DROP FUNCTION dbo.PythagoreanTriples;
GO
CREATE FUNCTION dbo.PythagoreanTriples (@srange, @erange)
RETURNS @hypotable TABLE (a INT, b INT, c INT)
AS
BEGIN
DECLARE @s1 INT, @s2 INT, @s3 INT;
SET @s1 = @srange;
BEGIN
SET @s2 = @erange;
WHILE (@s2 <= @erange)
BEGIN
SET @s3 = @srange;
WHILE (@s3 <= @erange)
BEGIN
IF (POWER(@s1, 2) + POWER(@s2, 2) = POWER(@s3, 2))
INSERT INTO @hypotable VALUES (@s1, @s2, @s3);
SET @s3 = @s3 + 1;
END;
SET @s2 = @s2 + 1;
END;
SET @s1 = @s1 + 1;
END;
RETURN
END;
GO
```

Task -2: Consider your database

Write user-defined functions

1. A function that returns the list of tables in your database.

```
IF OBJECT_ID ('dbo.getTableList') is not null drop function dbo.getTableList;
GO
Create function dbo.getTableList()
Returns table
As
Return
Select * From sys.objects
Where type = 'U'
GO
Select * from dbo.getTableList()
```

2. A function that returns all Primary-keys and Foreign-Keys.

```
IF OBJECT_ID ('dbo.getPKFK') is not null drop function dbo.getPKFK;
GO
Create function dbo.getPKFK()
Returns table
As
Return
select i.table_name, i.column_name, o.type
from sys.objects as o, information_schema.key_column_usage as i
where i.constraint_name = o.name and (o.type = 'PK' or o.type = 'F')
GO
Select * from dbo.getPKFK()
```

3. A function that returns all user-defined triggers.

```
IF OBJECT_ID ('dbo.getUDtrigger') is not null drop function dbo.getUDtrigger;
GO
Create function dbo.getUDtrigger()
Returns table
As
Return
Select type, name
From sys.objects
Where type = 'tr'
GO
Select * from dbo.getUDtrigger()
```

4. A function that returns row-count for all user-defined tables in the database.

```
IF OBJECT_ID ('dbo.getRCtable') is not null drop function dbo.getRCtable;
GO
Create function dbo.getRCtable ()
Returns table
As
Return
Select t.name, s.row_count
From sys.tables t, sys.dm_db_partition_stats s
Where t.object_id = s.object_id and t.type_desc = 'USER_TABLE'
GO
Select * from dbo.getRCtable()
```

5. A function that returns the list of all user defined functions.

```
IF OBJECT_ID ('dbo.getFunctionList') is not null drop function dbo.getFunctionList;
GO
Create function dbo.getFunctionList ()
Returns table
As
Return
Select name as function_name, schema_name(schema_id) as schema_name, type, type_desc
From sys.objects
Where type_desc like'%function%'
GO
Select * from dbo.getFunctionList()
```

6. A function that takes a table name as input and returns the column names with their types.

```
IF OBJECT_ID ('dbo.getColumnList') is not null drop function dbo.getColumnList;
Create function dbo.getColumnList (@cname varchar(40))
Returns table
As
Return
select schema_name (schema_id) as schema_name, o.name as object_name, o.type, o.type_desc,
c.column_id, c.name as column_name, type_name(c.user_type_id) as column_type, c.max_length
from sys.object as o, sys.columns as c
where o.object_id = c.object_id and o.object_id = OBJECT_ID(@cname)
go
select * from dbo.getColumnList('Instructor')
```

7. A function that takes a function name as input and returns input parameters for that function with their types.

```
IF OBJECT_ID ('dbo.getParameters') is not null drop function dbo.getParameters;
Create function dbo.getParameters (@fname varchar(40))
Returns table
As
Return
select schema_name (schema_id) as schema_name, o.name as object_name, o.type, o.type_desc,
p.parameter_id, p.name as parameter_name, type_name(p.user_type_id) as parameter_type,
p.max_length, p.precision
from sys.object as o, sys.parameters as p
where o.object_id = p.object_id and o.object_id = OBJECT_ID(@fname)
go
select * from dbo.getParameters('getColumnList')
```

Task -3: Consider your database project application

Define at least five functions related to your database project application and implement them by writing user defined functions.

```
Use LectureDB;
Go

--1. If you enter the name of a course, the room number of that course is printed out.
강의이름 입력하면 해당 강의의 강의실번호 출력
Create function getNumLR(@lName)
Returns table
As
Return
Select l.L_name as Lecture name, lr.LR_id as Lecture Room Num
From dbo.Lecture l, dbo.LectureRoom lr
Where l.LR_id = lr.LR_id and l.L_name = @lName
Go

--2. If you enter the discount coupon number, the course name of the coupon is printed.
할인 쿠폰 번호를 입력하면 해당 쿠폰의 강의이름 출력
Create function getNameLec(@dcNum)
Returns table
As
Return
Select dc.DC_id as Discount Coupon Num, l.L_name as Lecture name
From dbo.DiscountCoupon dc, dbo.Lecture l
Where dc.DC_id = @dcNum and dc.L_id = l.L_id
Go

--3. If you enter a course name, the instructor for that course will be printed.
강의이름 입력하면 해당 강의의 강사 출력
Create function getInst(@lName)
Returns table
As
Return
Select l.L_name as Lecture name, i.I_name as Instructor
From dbo.Lecture l, dbo.Instructor i
Where l.L_name = @lName and l.I_id = i.I_id
Go
```

```
--4. If you enter a classroom room, the maximum occupancy of the classroom is printed out.
강의실 번호 입력하면 해당 강의실 최대 수용인원 출력
Create function getNumMax(@lrId)
Returns table
As
Return
Select LR_id as Lecture Room Num, LR_maximumstudent as Maximum Capacity
From dbo.LectureRoom
Where LR_id = @lrId
Go

--5. If you enter the student's name and the student's phone number will be printed.
학생이름 입력하면 해당 학생의 전화번호 출력
Create function getPhoneNum(@sName)
Returns table
As
Return
Select S_name as Student Name, S_phone as Student Phone Number
From dbo.Student
Where S_name = @sName
Go
```

Task -4: Consider your database

Write store procedures for each table to 1. insert a record, 2. delete a record, 3. update a record.

Insert a record

```
Create procedure insert_LectureRoom(
@c1 varchar(10),
@c2 varchar(50),
@c3 int)
As
Begin
Declare @i = count(LR_id) from dbo.LectureRoom where LR_id = @c1;
If (@i > 0)
Begin
Raiserror('Error', 16, 1 )
End
Else
Begin
Insert into LectureRoom values(@c1, @c2, @c3)
End
End
Go
-----------------------------------------------------
Create procedure insert_Instructor(
@c1 varchar(10),
@c2 varchar(30),
@c3 varchar(10),
@c4 varchar(16),
@c5 varchar(40),
@c6 varchar(2000),
@c7 varchar(50))
As
Begin
Declare @i = count(I_id) from dbo.Instructor where I_id = @c1;
If (@i > 0)
Begin
Raiserror('Error', 16, 1 )
End
Else
```

```
Begin
Insert into Instructor values(@c1, @c2, @c3, @c4, @c5, @c6, @c7)
End
End
Go
-----------------------------------------------------
Create procedure insert_Lecture(
@c1 varchar(10),
@c2 varchar(30),
@c3 int,
@c4 date,
@c5 date,
@c6 int,
@c7 varchar(10),
@c8 time,
@c9 time,
@c10 int,
@c11 varchar(2000),
@c12 varchar(10),
@c13 varchar(10),
@c14 date)
As
Begin
Declare @i = count(L_id) from dbo.Lecture where L_id = @c1;
If (@i > 0)
Begin
Raiserror('Error', 16, 1 )
End
Else
Begin
Insert into Lecture values(@c1, @c2, @c3, @c4, @c5, @c6, @c7, @c8, @c9, @c10, @c11, @c12,
@c13, @c14)
End
End
Go
-----------------------------------------------------
Create procedure insert_Student(
@c1 varchar(10),
@c2 varchar(30),
```

```
@c3 date,
@c4 varchar(16))
As
Begin
Declare @i = count(S_id) from dbo.Student where S_id = @c1;
If (@i > 0)
Begin
Raiserror('Error', 16, 1 )
End
Else
Begin
Insert into Student values(@c1, @c2, @c3, @c4)
End
End
Go
-----------------------------------------------------
Create procedure insert_DiscountCoupon(
@c1 int,
@c2 int,
@c3 date,
@c4 varchar(10),
@c5 date)
As
Begin
Declare @i = count(DC_id) from dbo.DiscountCoupon where DC_id = @c1;
If (@i > 0)
Begin
Raiserror('Error', 16, 1 )
End
Else
Begin
Insert into DiscountCoupon values(@c1, @c2, @c3, @c4, @c5)
End
End
Go
-----------------------------------------------------
Create procedure insert_Take(
@c1 varchar(10),
@c2 varchar(10),
```

```
@c3 date)
As
Begin
Declare @i = count(S_id) from dbo.Take where S_id = @c1;
Declare @j = count(L_id) from dbo.Take where L_id = @c2;
If (@i > 0 || @j > 0 || @i <> @j)
Begin
Raiserror('Error', 16, 1 )
End
Else
Begin
Insert into Take values(@c1, @c2, @c3)
End
End
Go
-----------------------------------------------------
Create procedure insert_Possess(
@c1 int,
@c2 varchar(10),
@c3 date)
As
Begin
Declare @i = count(DC_id) from dbo.Possess where DC_id = @c1;
Declare @j = count(S_id) from dbo.Possess where S_id = @c2;
If (@i > 0 || @j > 0 || @i <> @j)
Begin
Raiserror('Error', 16, 1 )
End
Else
Begin
Insert into Possess values(@c1, @c2, @c3)
End
End
Go
```

Delete a record

```
Create procedure delete_LectureRoom(
@c1 varchar(10),
@c2 varchar(50),
@c3 int)
```

```
As
Begin
delete LectureRoom where c1 = @c1;
End
Go
-----------------------------------------------------
Create procedure delete_Instructor(
@c1 varchar(10),
@c2 varchar(30),
@c3 varchar(10),
@c4 varchar(16),
@c5 varchar(40),
@c6 varchar(2000),
@c7 varchar(50))
As
Begin
delete Instructor where c1 = @c1;
End
Go
-----------------------------------------------------
Create procedure delete_Lecture(
@c1 varchar(10),
@c2 varchar(30),
@c3 int,
@c4 date,
@c5 date,
@c6 int,
@c7 varchar(10),
@c8 time,
@c9 time,
@c10 int,
@c11 varchar(2000),
@c12 varchar(10),
@c13 varchar(10),
@c14 date)
As
Begin
delete Lecture where c1 = @c1;
End
```

```
Go
-----------------------------------------------------
Create procedure delete_Student(
@c1 varchar(10),
@c2 varchar(30),
@c3 date,
@c4 varchar(16))
As
Begin
delete Student where c1 = @c1;
End
Go
-----------------------------------------------------
Create procedure delete_DiscountCoupon(
@c1 int,
@c2 int,
@c3 date,
@c4 varchar(10),
@c5 date)
As
Begin
delete DiscountCoupon where c1 = @c1;
End
Go
-----------------------------------------------------
Create procedure delete_Take(
@c1 varchar(10),
@c2 varchar(10),
@c3 date)
As
Begin
delete Take where c1 = @c1;
End
Go
-----------------------------------------------------
Create procedure delete_Possess(
@c1 int,
@c2 varchar(10),
@c3 date)
```

```
As
Begin
delete Possess where c1 = @c1;
End
Go
```

Update a record

```
Create procedure update_LectureRoom(
@c1 varchar(10),
@c2 varchar(50),
@c3 int)
As
Begin
Update LectureRoom SET c1 = @c1;
End
Go
----------------------------------------------------------
Create procedure update_Instructor(
@c1 varchar(10),
@c2 varchar(30),
@c3 varchar(10),
@c4 varchar(16),
@c5 varchar(40),
@c6 varchar(2000),
@c7 varchar(50))
As
Begin
Update Instructor SET c1 = @c1;
End
Go
----------------------------------------------------------
Create procedure update_Lecture(
@c1 varchar(10),
@c2 varchar(30),
@c3 int,
@c4 date,
@c5 date,
@c6 int,
@c7 varchar(10),
@c8 time,
```

```
@c9 time,
@c10 int,
@c11 varchar(2000),
@c12 varchar(10),
@c13 varchar(10),
@c14 date)
As
Begin
Update Lecture SET c1 = @c1;
End
Go
-----------------------------------------------------
Create procedure update_Student(
@c1 varchar(10),
@c2 varchar(30),
@c3 date,
@c4 varchar(16))
As
Begin
Update Student SET c1 = @c1;
End
Go
-----------------------------------------------------
Create procedure update_DiscountCoupon(
@c1 int,
@c2 int,
@c3 date,
@c4 varchar(10),
@c5 date)
As
Begin
Update DiscountCoupon SET c1 = @c1;
End
Go
-----------------------------------------------------
Create procedure update_Take(
@c1 varchar(10),
@c2 varchar(10),
@c3 date)
```

```
As
Begin
Update Take SET c1 = @c1;
End
Go
-----------------------------------------------------
Create procedure update_Possess(
@c1 int,
@c2 varchar(10),
@c3 date)
As
Begin
Update Possess SET c1 = @c1;
End
Go
```