Amirkabir University of Technology (AUT)
Tehran, Iran,
Lab Database, Computer Engineering Department
Tina Gholami (9531307) Report 4

Amirkabir University of Technology
(Tehran Polytechnic)

Question 1)

تمرین۱: رویه ای بنویسید که وضعیت نمره ای کلاس را با بررسی جدول tblstudent که شامل فیلدهای نام دانشجو(stuName)، شماره دانشجو(stuID) و نمره ی دانشجو(stuGrade) است به عنوان خروجی برگرداند. اگر تعداد دانشجویانی که نمره ی کم‌تر از ۱۰ گرفته اند، حداکثر یک نفر باشد وضعیت خوب(GOOD) است. اگر این تعداد دو یا سه نفر باشد وضعیت کلاس نرمال(Normal) است و در نهایت اگر این تعداد بیش تر بود، وضعیت کلاس بد(Bad) است.

توجه: شماره ی دانشجویی کلید اصلی است. و stuGrade از نوع real است.

As mentioned above, we first have to create the table and insert some data into it:

```
create table tblstudent(
    stuID int not null primary key,
    stuName varchar(50),
    stuGrade real
);
```

```
insert into tblstudent(stuID, stuName, stuGrade) values (9531300, 'Gino Masaoka', 4);
insert into tblstudent(stuID, stuName, stuGrade) values (9531301, 'Sherlock Holms', 19.72);
insert into tblstudent(stuID, stuName, stuGrade) values (9531302, 'Shogo Makishima', 0);
insert into tblstudent(stuID, stuName, stuGrade) values (9531303, 'James Moriarty', 20);
insert into tblstudent(stuID, stuName, stuGrade) values (9531304, 'Holdon Calfield', 1.3);
insert into tblstudent(stuID, stuName, stuGrade) values (9531305, 'Sam Smith', 10);
insert into tblstudent(stuID, stuName, stuGrade) values (9531306, 'Patric Egan', 8.6);
insert into tblstudent(stuID, stuName, stuGrade) values (9531307, 'Tina Gholami', 17.2);
insert into tblstudent(stuID, stuName, stuGrade) values (9531308, 'Allison Pert', 2.2);
insert into tblstudent(stuID, stuName, stuGrade) values (9531309, 'Negin Soleymani', 12);
insert into tblstudent(stuID, stuName, stuGrade) values (9531310, 'Melody amelodir', 6.7);
insert into tblstudent(stuID, stuName, stuGrade) values (9531311, 'Ali Ebadi', 3.5);
insert into tblstudent(stuID, stuName, stuGrade) values (9531312, 'Alireza fm', 0.75);
```

	stuID	stuName	stuGrade
1	9531300	Gino Masaoka	4
2	9531301	Sherlock Holms	19.72
3	9531302	Shogo Makishima	0
4	9531303	James Moriarty	20
5	9531304	Holdon Calfield	1.3
6	9531305	Sam Smith	10
7	9531306	Patric Egan	8.6
8	9531307	Tina Gholami	17.2
9	9531308	Allison Pert	2.2
10	9531309	Negin Soleymani	12
11	9531310	Melody amelodir	6.7
12	9531311	Ali Ebadi	3.5
13	9531312	Alireza fm	0.75

Next, the desired procedure is created within the query:

```
create procedure StuStatus
as
declare @num_stu int;
select @num_stu = count(stuID) from tblstudent
where stuGrade < 10;

if @num_stu <= 1
print 'GOOD';
else if @num_stu between 2 and 3
print 'Normal';
else
print 'Bad';


exec StuStatus;
```

And here is the output:

```
Bad

Completion time: 2021-04-17T22:02:28.2881444+04:30
```

Question 2)

تمرین۲. روالی بنویسید که پارامتر num از نوع int را از ورودی بگیرد، با تعداد نمرات زیر ۱۰ از tblStudent مقایسه کند، اگر تعداد نمرات زیر ده از num کمتر بود، نمرات بین ۹ و ۱۰ را یک نمره اضافه کند، در غیر این صورت نمرات بین ۹,۵ تا ۱۰ را ۰,۵ اضافه کند.

The procedure is created as the following:

```
create procedure add_grade @num int
as
declare @less_than_10 int;
select @less_than_10 = count(stuGrade) from tblstudent
where stuGrade < 10;

if @less_than_10 < @num
update tblstudent
set stuGrade = stuGrade + 1
where stuGrade between 9 and 10

else
update tblstudent
set stuGrade = stuGrade + 0.5
where stuGrade between 9.5 and 10;


exec add_grade @num = 3;
```

Since @num = 3, and according to the table created in (Question 1) @less_than_10 = 6, then the "else" part will only be executed since @less_than_10 > @num.

Also, since we only have grade "10" for "Sam Smith" that is within the range of 9.5 and 10, then only this grade will be incremented by 0.5. Hence, only "Sam Smith" grade will be changed from 10 to 10.5.

Here is the output:

	stuID	stuName	stuGrade
1	9531300	Gino Masaoka	4
2	9531301	Sherlock Holms	19.72
3	9531302	Shogo Makishima	0
4	9531303	James Moriarty	20
5	9531304	Holdon Calfield	1.3
6	9531305	Sam Smith	10.5
7	9531306	Patric Egan	8.6
8	9531307	Tina Gholami	17.2
9	9531308	Allison Pert	2.2
10	9531309	Negin Soleymani	12
11	9531310	Melody amelodir	6.7
12	9531311	Ali Ebadi	3.5
13	9531312	Alireza fm	0.75

Question 3)

تمرین۳. روالی بنویسید که دو عدد را swap کند. روال را با اعداد ۱۲۳ و ۲۴ امتحان کنید .

In a procedure, we have to define both inputs and outputs to swap the two numbers and print them at the end.

Here is the procedure:

```
create procedure swap @num_1 int, @num_2 int, @out_1 int output, @out_2 int output
as
set @out_1 = @num_2
set @out_2 = @num_1
;


declare @number_1 int, @number_2 int;
exec swap @num_1 = 123, @num_2 = 24, @out_1 = @number_1 output, @out_2 = @number_2 output;
print @number_1;
print @number_2;
```

As can be seen, @num_1 = 123, and @num_2 = 24. Hence after swapping, @number_1 should be 24 and @number_2 should be 123. We have printed both and here are the results:

```
24
123

Completion time: 2021-04-17T22:35:59.2174704+04:30
```

As can be seen above, @number_1 = 24, and @number_2 = 123; as expected.

Question 4)

تمرین۴. تابعی بنویسید که نام یک دانشجو را به عنوان ورودی بگیرد و نمره‌ی آن دانشجوی خاص را برگرداند. دانشجویی با نام ali با شماره‌ی دانشجویی 9012345 و نمره ی ۱۴ را وارد جدول کنید. تابع نوشته شده را با 'ali' امتحان کنید.

First, new student "ali" is inserted into the database relation:

```
insert into tblstudent values (9012345, 'ali', 14);
```

And here is how the relation will look like after adding "ali" to the table:

	stuID	stuName	stuGrade
1	9012345	ali	14
2	9531300	Gino Masaoka	4
3	9531301	Sherlock Holms	19.72
4	9531302	Shogo Makishima	0
5	9531303	James Moriarty	20
6	9531304	Holdon Calfield	1.3
7	9531305	Sam Smith	10
8	9531306	Patric Egan	8.6
9	9531307	Tina Gholami	17.2
10	9531308	Allison Pert	2.2
11	9531309	Negin Soleymani	12
12	9531310	Melody amelodir	6.7
13	9531311	Ali Ebadi	3.5
14	9531312	Alireza fm	0.75

Next, function "Grade" is created. This function receives a student name (@name) and returns his/her grade (@temp) as an integer value. Then name "ali" is fed to the function as input, and his grade (which was defined to be 14) will be returned and assigned to variable "@grade_student". Finally, we will print the result (14).

Here is the function “Grade”:

```
create function Grade(@name varchar(50))
returns int
as
begin
    declare @temp int
    select @temp = stuGrade from tblstudent where stuName = 'ali'
    return @temp;
end



declare @grade_student int;
exec @grade_student = Grade @name = 'ali'
print @grade_student;
```

And the output is:

```
14

Completion time: 2021-04-17T22:52:43.6172923+04:30
```

Question 5)

تمرین۵. جدول زیر را وارد کرده و با استفاده از توابع رنکدهی گفته شده، بر اساس postal code رنک دهی کنید.

First the table is created:

```
create table People(
    FirstName varchar(20),
    LastName varchar(20),
    PostalCode int
);

insert into People(FirstName, LastName, PostalCode) values('Michael', 'Blythe', 98027);
insert into People(FirstName, LastName, PostalCode) values('Linda', 'Mitchell', 98027);
insert into People(FirstName, LastName, PostalCode) values('Jillian', 'Carson', 98027);
insert into People(FirstName, LastName, PostalCode) values('Garrett', 'Vargas', 98027);
insert into People(FirstName, LastName, PostalCode) values('Tsvi', 'Reiter', 98027);
insert into People(FirstName, LastName, PostalCode) values('Shu', 'Ito', 98055);
insert into People(FirstName, LastName, PostalCode) values('Jose', 'Saraiva', 98055);
insert into People(FirstName, LastName, PostalCode) values('David', 'Campbell', 98055);
insert into People(FirstName, LastName, PostalCode) values('Tete', 'Mensa-Annan', 98055);
insert into People(FirstName, LastName, PostalCode) values('Lynn', 'Tsoflias', 98055);
insert into People(FirstName, LastName, PostalCode) values('Rachel', 'Valdez', 98055);
insert into People(FirstName, LastName, PostalCode) values('Jae', 'Pak', 98055);
insert into People(FirstName, LastName, PostalCode) values('Ranjit', 'Varkey Chudukatil', 98055);
```

	FirstName	LastName	PostalCode
1	Michael	Blythe	98027
2	Linda	Mitchell	98027
3	Jillian	Carson	98027
4	Garrett	Vargas	98027
5	Tsvi	Reiter	98027
6	Shu	Ito	98055
7	Jose	Saraiva	98055
8	David	Campbell	98055
9	Tete	Mensa-Annan	98055
10	Lynn	Tsoflias	98055
11	Rachel	Valdez	98055
12	Jae	Pak	98055
13	Ranjit	Varkey Chudukatil	98055

Here are the three ranking functions (row_number, rank, dense_rank) applied to the relation:

```
select *,
    ROW_NUMBER() over (order by PostalCode) as RowNumber,
    RANK() over (order by PostalCode) as RankNumber,
    DENSE_RANK() over (order by PostalCode) as DenseRank
from People;
```

And the output is:

	FirstName	LastName	PostalCode	RowNumber	RankNumber	DenseRank
1	Michael	Blythe	98027	1	1	1
2	Linda	Mitchell	98027	2	1	1
3	Jillian	Carson	98027	3	1	1
4	Garrett	Vargas	98027	4	1	1
5	Tsvi	Reiter	98027	5	1	1
6	Shu	Ito	98055	6	6	2
7	Jose	Saraiva	98055	7	6	2
8	David	Campbell	98055	8	6	2
9	Tete	Mensa-...	98055	9	6	2
10	Lynn	Tsoflias	98055	10	6	2
11	Rachel	Valdez	98055	11	6	2
12	Jae	Pak	98055	12	6	2
13	Ranjit	Varkey ...	98055	13	6	2

As can be seen, rows are ranked based on the ranking functions applied.

Question 6)

تمرین ۶. تابعی بنویسید که زمان جاری را به عنوان ورودی بگیرد و و قسمت روز ۴ روز بعد را به عنوان خروجی برگرداند.(راهنمایی: استفاده از DATENAME).

Here is the function to receive today's date (which was "Apr 18 2021 2:29AM" at the time of execution) as input and returns the day part of the next four days as output:

```
create function four_days_later(@input varchar(50))
returns varchar(50)
as
begin
    declare @temp varchar(50);
    select @temp = DATEADD(day, 4, @input);
    select @temp = DATENAME(day, @temp);
    --declare @temp2 varchar(50);
    --select @temp2 = DATENAME(day, @temp)
    --return @temp2;
    return @temp;
end


declare @output varchar(50);
declare @time varchar(50);
select @time = GETDATE();
exec @output = four_days_later @input = @time;
print @output;
```

Here is the input:

```
Apr 18 2021  2:29AM

Completion time: 2021-04-18T02:29:22.6812264+04:30
```

Here is the output:

```
22

Completion time: 2021-04-18T02:28:06.6052588+04:30
```

Question 7)

تمرین ۷. آیا function ها را می توان با دستوراتی مانند insert/update/delete به کار برد(امتحان کنید و نتیجه را همراه با عکس ذکر کنید).

No, we cannot do Insert/Update/Delete. Functions only work with select statements. They have only READ-ONLY database access. If DML operations would be allowed in functions, then function would be pretty similar to stored Procedure. Basically, this is what makes functions different from Procedures. If we wish to perform data change, we can just create a Stored Procedure for that. And according to the SQL documentation for functions: "User-defined functions cannot be used to perform actions that modify the database state."

Here, I have tried the "delete" query inside function "Grade" on the previously built table "People" to delete the instances with FirstName = 'Michael':

```
create function four_days_later(@input varchar(50))
returns varchar(50)
as
begin
    declare @temp varchar(50);
    select @temp = DATEADD(day, 4, @input);
    select @temp = DATENAME(day, @temp);
    delete from People where FirstName = 'Michael';
    return @temp;
end
```

As can be seen, we will get error while using "delete" inside a SQL function since it is not allowed to use "delete" inside a function.

```
Msg 443, Level 16, State 15, Procedure four_days_later, Line 8 [Batch Start Line 0]
Invalid use of a side-effecting operator 'DELETE' within a function.

Completion time: 2021-04-18T15:28:39.5297572+04:30
```

Next, "Update" query was used within the "Grade" function to update the previously built table "People":

```
create function four_days_later(@input varchar(50))
returns varchar(50)
as
begin
    declare @temp varchar(50);
    select @temp = DATEADD(day, 4, @input);
    select @temp = DATENAME(day, @temp);
    update People
    set PostalCode = 1111
    where FirstName = 'Michael';
    return @temp;
end
```

As can be seen, there is error for using “Update” inside a Function:

```
Msg 443, Level 16, State 15, Procedure four_days_later, Line 8 [Batch Start Line 0]
Invalid use of a side-effecting operator 'UPDATE' within a function.

Completion time: 2021-04-18T15:25:09.4150594+04:30
```

And finally, “insert” is checked:

```
create table Hello(
    name varchar(20),
    age int
);


create function four_days_later(@input varchar(50))
returns varchar(50)
as
begin
    declare @temp varchar(50);
    select @temp = DATEADD(day, 4, @input);
    select @temp = DATENAME(day, @temp);
    insert into Hello(name, age) values('Tina', 23);
    return @temp;
end
```

And as expected, we will face error while using “insert” inside a SQL function “Grade”:

```
Msg 443, Level 16, State 15, Procedure four_days_later, Line 14 [Batch Start Line 0]
Invalid use of a side-effecting operator 'INSERT' within a function.

Completion time: 2021-04-18T15:36:07.4832060+04:30
```

Question 8)

Search about the differences between a SQL Function and Procedure.

Stored Procedures are pre-compiled objects which are compiled for the first time and its compiled format is saved, which executes (compiled code) whenever it is called. A function is compiled and executed every time whenever it is called. A function must return a value and cannot modify the data received as parameters. The main differences are:

1. The function must return a value but in Stored Procedure it is optional. Even a procedure can return zero or n values.
2. Functions can have only input parameters for it whereas Procedures can have input or output parameters.
3. Functions can be called from Procedure whereas Procedures cannot be called from a Function.
4. The procedure allows SELECT as well as DML(INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.
5. Procedures cannot be utilized in a SELECT statement whereas Function can be embedded in a SELECT statement.
6. Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be.
7. Functions that return tables can be treated as another row set. This can be used in JOINs with other tables.
8. Inline Function can be thought of as views that take parameters and can be used in JOINs and other row set operations.
9. An exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.
10. We can use Transactions in Procedure whereas we can't use Transactions in Function.