Function

(Лабораторн №5)

У. Төрболд

ХШУИС, Мэдээллийн технологийн IV-р түвшний оюутан, turbold1125@gmail.com

1. ОРШИЛ

Энэхүү лабораторид SQL – ийн функцууд буюу тодорхой хэрэглэгчийн тодорхойлсон функцийг ашиглан кодыг модульчлах, дахин ашиглах, урьдчилан бэлдсэн код дуудсанаар өгөгдлийн сангийн гүйцэтгэлийг сайжруулна. Энэхүү лабораторид SQL функцуудын тухай ойлголт, үүнийг практик хэрэглээнд хэрхэн ашиглах талаар судлах болно.

2. ЗОРИЛГО

Энэхүү лабораторын зорилго function – ын талаар практик дээр хэрэгжүүлэх бөгөөд үүнийг хэрэгжүүлснээр кодын зохион байгуулалт, дахин ашиглах, гүйцэтгэлийг сайжруулах боломжтой болно. Үүний тулд дараах зорилтуудыг хэрэгжүүлэхийг зорьлоо.

- Борлуулалтын мэдээлэлд үндэслэн Nothwind database – д функц бичих. Энэ функц нь бүтээгдэхүүн бүрийн захиалах хэмжээг урьдчилан таамаглах замаар бүтээгдэхүүн менежмент-д туслах зорилготой.
- Сургуулийн мэдээллийн санд зориулан GPA буюу голч дүнг тооцоолох функц бичих. Энэ функц нь урьдчилан тодорхойлсон томъёо ашиглан тухайн оюутны голч оноог тооцоолох зорилготой.
- Хүн ам бүртгэлийн мэдээлэлд үндэслэн насаар нь ангилах функц бичих. Энэ функц нь тухайн хотод хэдэн хүн байгаа тоог харуулах, насаар нь харуулах зэргээр хүн амын нягтаршилыг тооцоолох зорилготой.

3. ОНОЛЫН СУДАЛГАА

3.1 Function

SQL функцууд нь параметрүүдийг хүлээн авах, тооцоолол хийх, нэг утгыг буцаах, кодыг дахин ашиглах боломжтой болгодог. Үүний скаляр болон хүснэгтийн функц гэж хоёр төрөлд хуваж болно. Скаляр функцууд нь бүхэл тоо, мөр, огноо гэх мэт нэг утгыг буцаана. Тэдгээрийг ихэвчлэн тооцоолол, мөрийг удирдах, огнооны үйлдлүүдэд ашигладаг. Хүснэгтийн функц нь хүснэгт хэлбэрээр үр дүнг буцаадаг. Тэд параметрүүдийг хүлээн авч, динамик үр дүнг үүсгэхийн тулд нарийн төвөгтэй байдаг.

Syntax:

```
- Transact-SQL Scalar Function Syntax
CREATE [ OR ALTER ] FUNCTION [ schema_name. ] function_name
```

```
( [ { @parameter_name [ AS ][ type_schema_name. ] parameter_data_type [ NULL ]
 [ = default ] [ READONLY ] }
    [ ,...n ]
  ]
)
RETURNS return_data_type
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    BEGIN
        function_body
        RETURN scalar_expression
    END
[ ; ]
```

- **function_name** : Энэ нь заасан өгөгдлийн санд 'function_name' гэсэн скаляр функцийг үүсгэнэ. OR ALTER нь функц байгаа бол өөрчлөх боломжийг олгодог.
- **([{ @parameter_name [AS][type_schema_name.] parameter_data_type [NULL] [= default] [READONLY] } :** Энэ нь функцийг оролтын параметрүүдийг тодорхойлдог. Параметр бүрийг **'@parameter_name'**, өгөгдлийн төрөл **'parameter_data_type'** болон **NULL**, анхдагч утга буюу **READONLY** зэрэг шинж чанаруудыг зааж өгдөг.
- **RETURNS return_data_type :** Энэ нь функцийг гүйцэтгэсний дараа буцаах өгөгдлийн төрлийг заана.
- **[WITH <function_option> [,...n]] :** Энэ нь шифрлэлт, холболт зэрэг функцэд зориулсан төрөл бүрийн сонголтуудыг тодорхойлно. Жишээ нь хувьсагч зарлах гэх мэт
- **[AS] :** Энэ нь функцийн бие эхэлж буйг заана.
- **BEGIN :** Функцийн процедурын логикийн эхлэлийг тэмдэглэнэ.
- **function_body :** Функцийн гүйцэтгэх логик буюу тооцооллыг агуулна. Функцийн үйлдлийг тодоройлсон кодыг бичнэ.
- **RETURN scalar_expression :** Функцийн буцаах утгыг заана. '**scalar_expression'** нь хувьсагч, багана эсвэл скаляр функц байж болно.
- **END :** Функцийн процедурын логикийн төгсгөлийг тэмдэглэнэ.

4. ХЭРЭГЖҮҮЛЭЛТ

4.1 Дараа сард нийлүүлэх тоог тооцох. Өмнөх 3 сарын (хамгийн сүүлд бүртгэгдсэн огноогоос өмнөх 3 сар гэсэн үг) барааны борлуулалтын дундаж тоог мөн одоо байгаа барааны үлдэгдлийг тооцон дараа сард заахиалга өгөх дүнг харуулдаг функц бич. Барааны ID – г параметрээр өгнө.

Жишээ нь: 1 дугаартай product өмнөх 3 сарын дунджаар сард 50ш борлогдсон ба одоо үлдэгдэл 10ш байвал, нэмж 40ш захиалах шаардлагатай гэдгийг харуулна гэсэн үг.

```
-- Функц тодорхойлох, ProductID гараас авах
CREATE FUNCTION CalculateOrderAmountForNextMonth(@ProductID INT)
RETURNS INT
AS
BEGIN
       -- Дундаж борлуулалт, одоогийн үлдэгдэл, захиалгын дүнг хадгалах хувьсагч зарлах
    DECLARE @AverageSales INT;
    DECLARE @CurrentBalance INT;
    DECLARE @OrderAmountForNextMonth INT;
       -- Заасан бүтээгдэхүүний өмнөх 3 сарын дундаж борлуулалтыг тооцоолох
    SELECT @AverageSales = ROUND(AVG(Quantity), 0)
```

```
    FROM (
       -- Сүүлийн 3 сарыг багтаан сар бүрийн тоо хэмжээг нэгтгэх subquery
        SELECT TOP 3 SUM(Quantity) AS Quantity
        FROM [Order Details] od
        JOIN Orders o ON od.OrderID = o.OrderID
        WHERE od.ProductID = @ProductID
              -- 1998 оны 4-р сараас хойшхи захиалгыг оруулахгүй
              AND o.OrderDate < '1998-04-01'
        GROUP BY YEAR(o.OrderDate), MONTH(o.OrderDate)
        ORDER BY YEAR(o.OrderDate) DESC, MONTH(o.OrderDate) DESC
    ) AS AvgSales;
       -- Product table - с тухайн бүтээгдэхүүний одоогийн үлдэгдлийг хадгалах
    SELECT @CurrentBalance = UnitsInStock
    FROM Products
    WHERE ProductID = @ProductID;
       -- Дундаж борлуулалт болон одоогийн үлдэгдэл дээр үндэслэн дараагийн сарын захиалгын
хэмжээг тооцоолох
    SET @OrderAmountForNextMonth = @AverageSales - @CurrentBalance;
       -- Захиалгын дүн сөрөг биш эсэх
    IF @OrderAmountForNextMonth < 0
    BEGIN
        SET @OrderAmountForNextMonth = 0;
    END;
       -- Дараагийн сард тооцоолсон захиалгын дүнг буцаана
    RETURN @OrderAmountForNextMonth;
END;
-- Функцийг дуудаж ProductID хувьсагчид 1 утгыг оноох.
SELECT dbo.CalculateOrderAmountForNextMonth(1) AS QuantityToDeliver;
```

Үр дүн:

```
SELECT dbo.CalculateOrderAmountForNextMonth(1) AS QuantityToDeliver;

SELECT ...;

SELECT UnitsInStock FROM Products WHERE ProductID = 1;
```

Results | Messages

	QuantityToDeliver
1	46

	Year	Month	TotalQuantity
1	1998	3	81
2	1998	2	90
3	1998	1	84

	UnitsInStock
1	39

Үүнд барааны үлдэгдэл 39ш. Функцыг дуудаж ProductID = 1 үед 46 бараа нэмж захиалах боломжтойг харуулж байна. Дундах үр дүн нь сар бүрийн дундаж бараа захиалгын үр дүнг харуулна.

4.2 CalculateGPA нэртэй оюутны дугаарыг өгөхөөр дүнгийн дундаж-ыг бодож харуулдаг функц бич.(дундаж дүн боддог томъёог ашигла.) Бичсэн функцээ ашиглан Бүх оюутны нэр, дундаж дүнг буурахаар эрэмбэлж харуул.

```
-- Функц зарлах, StudentID гараас авах
```

```sql
CREATE FUNCTION calculateGPA (@StudentID INT)
RETURNS DECIMAL(3, 2)
AS
BEGIN
    -- Нийт кредит болон нийт дүнг хадгалах хувьсагч
    DECLARE @TotalCredits INT
    DECLARE @TotalGradePoints DECIMAL(10, 2);
    DECLARE @GPA DECIMAL(3, 2);
   -- Оюутны нийт кредит болон нийт дүнгийн оноог тооцоолох
    SELECT @TotalCredits = SUM(Credits),
           @TotalGradePoints = SUM(Credits * Grade)
    FROM dbo.StudentGrade sg
    INNER JOIN dbo.Course c ON sg.CourseID = c.CourseID
    WHERE StudentID = @StudentID;
    -- Нийт кредит 0-ээс их бол голч оноог тооцоол
    IF @TotalCredits > 0
       BEGIN
             SET @GPA = @TotalGradePoints / @TotalCredits;
       END
    ELSE
       BEGIN
             SET @GPA = NULL;
       END
   -- Тооцоолсон голч дүнг буцаана
   RETURN @GPA;
END

-- Функцийг дуудаж studentID хувьсагчид 2 утгыг оноох.
SELECT dbo.calculateGPA(2) AS GPA;

-- Функцийг дуудаж буурах эрэмбээр жагсаах
SELECT FirstName + ' ' + LastName AS StudentName,
       dbo.calculateGPA(PersonID) AS GPA
FROM dbo.Person
WHERE Discriminator = 'Student'
   ORDER BY GPA DESC;
```

Үр дүн:

```sql
SELECT FirstName + ' ' + LastName AS StudentName,
       dbo.calculateGPA(PersonID) AS GPA
FROM dbo.Person
WHERE Discriminator = 'Student'
ORDER BY GPA DESC;

GO
```

82 % — Results | Messages

	StudentName	GPA
1	Arturo Anand	4.00
2	Robyn Suarez	4.00
3	Randall Martin	4.00
4	Rachel Griffin	4.00
5	Gytis Barzdukas	3.80
6	Laura Norman	3.79
7	Alicia Shan	3.75

```sql
SELECT dbo.calculateGPA(2) AS GPA;
GO

-- Функцийг дуудаж буурах эрэмбээр
```

82 % — Results | Messages

	GPA
1	3.80

```sql
select * from StudentGrade WHERE StudentID = 2
```

2 % — Results | Messages

	EnrollmentID	CourseID	StudentID	Grade
1	1	2021	2	4.00
2	2	2030	2	3.50

4.3 Хэрэглэгчийн функц ашиглан Тайлан гаргах. Уг дасгалыг 2 аргаар гүйцэтгэнэ.

1. Аймгийн дугаарыг өгөхөөр санд байгаа бүх насны хүмүүсээс тухайн аймгаас хэдэн хүн байгаа тоог гаргадаг функц бичээд бүх аймгуудын хувьд JOIN хийж тайланг гаргаж хугацааг тэмдэглэж авах.

```
-- Функц зарлах, гараас RegionID авах
CREATE FUNCTION PopulationCountRegion(@RegionID INT)
RETURNS TABLE
AS
RETURN
(
    -- RegionID таарч байгаа хэрэглэгчдийг тоолох
    SELECT COUNT(*) AS PopulationCount
    FROM Users
    WHERE RegionID = @RegionID
);

DECLARE @StartTime2 DATETIME, @EndTime2 DATETIME, @Method2ExecutionTime INT;
SET @StartTime2 = GETDATE();

-- Функц дуудах, Regions table – аас ID авах
SELECT
    r.Region_name AS RegionName,
    (
        SELECT PopulationCount
        FROM dbo.PopulationCountRegion(r.ID)
    ) AS PopulationCount
FROM
    Regions r;

SET @EndTime2 = GETDATE();
SET @Method2ExecutionTime = DATEDIFF(MILLISECOND, @StartTime2, @EndTime2);
```

Үр дүн:

	RegionName	PopulationCount
1	Ulaanbaatar	8
2	Darhan	8
3	Erdenet	4

	Method2ExecutionTime
1	0

2. Аймгийн дугаар, нас гэсэн 2 аргумент аваад тоог нь буцаадаг функц бичээд нүд бүрийн хувьд бодуулах замаар тайлан үүсгэх. Хугацааг тэмдэглэж аваад эхний аргатай харьцуулж үзэх

```
    -- Функц зарлах, гараас Age, RegionID авах
CREATE FUNCTION AgeReport(@Age INT, @RegionID INT)
RETURNS TABLE
AS
RETURN
```

```
(
       -- RegionID дахь хэрэглэгчийг тоолох
    SELECT COUNT(*) AS PopulationCount
    FROM Users
    WHERE RegionID = @RegionID AND DATEDIFF(YEAR, DOB, GETDATE()) = @Age
);


DECLARE @StartTime1 DATETIME, @EndTime1 DATETIME, @Method1ExecutionTime INT;
SET @StartTime1 = GETDATE();

-- Функц дуудах, Users table – аас DOB авч Age тооцоолон, параметрээр Age, RegionID
дамжуулах
SELECT
    Age,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 1)) AS Ulaanbaatar,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 2)) AS Darhan,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 3)) AS Erdenet
FROM
    (SELECT DISTINCT DATEDIFF(YEAR, DOB, GETDATE()) AS Age FROM Users) AS Age;

SET @EndTime1 = GETDATE();
SET @Method1ExecutionTime = DATEDIFF(MILLISECOND, @StartTime1, @EndTime1);

SELECT @Method1ExecutionTime AS Method1ExecutionTime;
```

Үр дүн:

	Age	Ulaanbaatar	Darhan	Erdenet
1	22	2	2	0
2	23	4	2	2
3	24	2	4	2

	Method1ExecutionTime
1	0

5. ДҮГНЭЛТ

SQL функц нь мэдээллийн санд логикыг багтаах, тооцоолол хийх хэрэгтэй функц юм. Эдгээр нь кодыг оновчтой болгох, засвар үйлчилгээ хийх, гүйцэтгэлийг сайжруулах арга замыг бий болгодог. Энэхүү тайланд дурьдсан функцуудыг хэрэгжүүлснээр тодорхой даалгаврыг автоматжуулж нарийн төвөгтэй үйлдлүүдийг хялбарчилна.

6. Хавсралт

6.1.

```
CREATE FUNCTION CalculateOrderAmountForNextMonth(@ProductID INT)
RETURNS INT
AS
BEGIN
    DECLARE @AverageSales INT;
    DECLARE @CurrentBalance INT;
    DECLARE @OrderAmountForNextMonth INT;
```

```
        SELECT @AverageSales = ROUND(AVG(Quantity), 0)
        FROM (
            SELECT TOP 3 SUM(Quantity) AS Quantity
            FROM [Order Details] od
            JOIN Orders o ON od.OrderID = o.OrderID
            WHERE od.ProductID = @ProductID
                  AND o.OrderDate < '1998-04-01'
            GROUP BY YEAR(o.OrderDate), MONTH(o.OrderDate)
            ORDER BY YEAR(o.OrderDate) DESC, MONTH(o.OrderDate) DESC
        ) AS AvgSales;

        SELECT @CurrentBalance = UnitsInStock
        FROM Products
        WHERE ProductID = @ProductID;

        SET @OrderAmountForNextMonth = @AverageSales - @CurrentBalance;

    IF @OrderAmountForNextMonth < 0
    BEGIN
        SET @OrderAmountForNextMonth = 0;
    END;

    RETURN @OrderAmountForNextMonth;
END;

SELECT dbo.CalculateOrderAmountForNextMonth(1) AS QuantityToDeliver;
```

6.2.

```
CREATE FUNCTION calculateGPA (@StudentID INT)
RETURNS DECIMAL(3, 2)
AS
BEGIN
    DECLARE @TotalCredits INT
    DECLARE @TotalGradePoints DECIMAL(10, 2);
   DECLARE @GPA DECIMAL(3, 2);

    SELECT @TotalCredits = SUM(Credits),
           @TotalGradePoints = SUM(Credits * Grade)
    FROM dbo.StudentGrade sg
    INNER JOIN dbo.Course c ON sg.CourseID = c.CourseID
    WHERE StudentID = @StudentID;

    IF @TotalCredits > 0
       BEGIN
              SET @GPA = @TotalGradePoints / @TotalCredits;
       END
   ELSE
       BEGIN
              SET @GPA = NULL;
       END
    RETURN @GPA;
END

SELECT dbo.calculateGPA(2) AS GPA;
```

6.3.

```
SELECT FirstName + ' ' + LastName AS StudentName,
          dbo.calculateGPA(PersonID) AS GPA
```

```
FROM dbo.Person
WHERE Discriminator = 'Student'
ORDER BY GPA DESC;
```

6.4.

```
CREATE FUNCTION PopulationCountRegion(@RegionID INT)
RETURNS TABLE
AS
RETURN
(
    SELECT COUNT(*) AS PopulationCount
    FROM Users
    WHERE RegionID = @RegionID
);

DECLARE @StartTime2 DATETIME, @EndTime2 DATETIME, @Method2ExecutionTime INT;
SET @StartTime2 = GETDATE();

SELECT
    r.Region_name AS RegionName,
    (
        SELECT SUM(PopulationCount)
        FROM dbo.PopulationCountRegion(r.ID)
    ) AS PopulationCount
FROM
    Regions r;

SET @EndTime2 = GETDATE();
SET @Method2ExecutionTime = DATEDIFF(MILLISECOND, @StartTime2, @EndTime2);
```

6.5.

```
CREATE FUNCTION AgeReport(@Age INT, @RegionID INT)
RETURNS TABLE
AS
RETURN
(
    SELECT COUNT(*) AS PopulationCount
    FROM Users
    WHERE RegionID = @RegionID AND DATEDIFF(YEAR, DOB, GETDATE()) = @Age
);

DECLARE @StartTime1 DATETIME, @EndTime1 DATETIME, @Method1ExecutionTime INT;
SET @StartTime1 = GETDATE();

SELECT
    Age,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 1)) AS Ulaanbaatar,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 2)) AS Darhan,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 3)) AS Erdenet
FROM
    (SELECT DISTINCT DATEDIFF(YEAR, DOB, GETDATE()) AS Age FROM Users) AS Age;

DECLARE @StartTime1 DATETIME, @EndTime1 DATETIME, @Method1ExecutionTime INT;
SET @StartTime1 = GETDATE();

SELECT @Method1ExecutionTime AS Method1ExecutionTime;
```