IS664 Database Programming
Fall 2022

Stored Functions

LECTURE 5: PROGRAMMING CONSTRUCTS FOR USER-DEFINED FUNCTIONS

General

▶ The programming power of MySQL **is limited** when compared to other languages.

▶ MySQL language constructs are **designed specifically to work with MySQL databases** rather than as a general-purpose programming language.

▶ MySQL provided **extensions to SQL** known as stored programs. **Stored programs** can include procedural code that controls the flow of execution of a database operation.

▶ **There are four types of stored programs:**

  ▶ **Stored Procedure**

    ▶ Can be called from an application that has access to the database.

  ▶ **Stored Function**

    ▶ Can be called from a SQL statement.

  ▶ **Trigger**

    ▶ Is executed in response to an INSERT, UPDATE, or DELETE statement on a specific table.
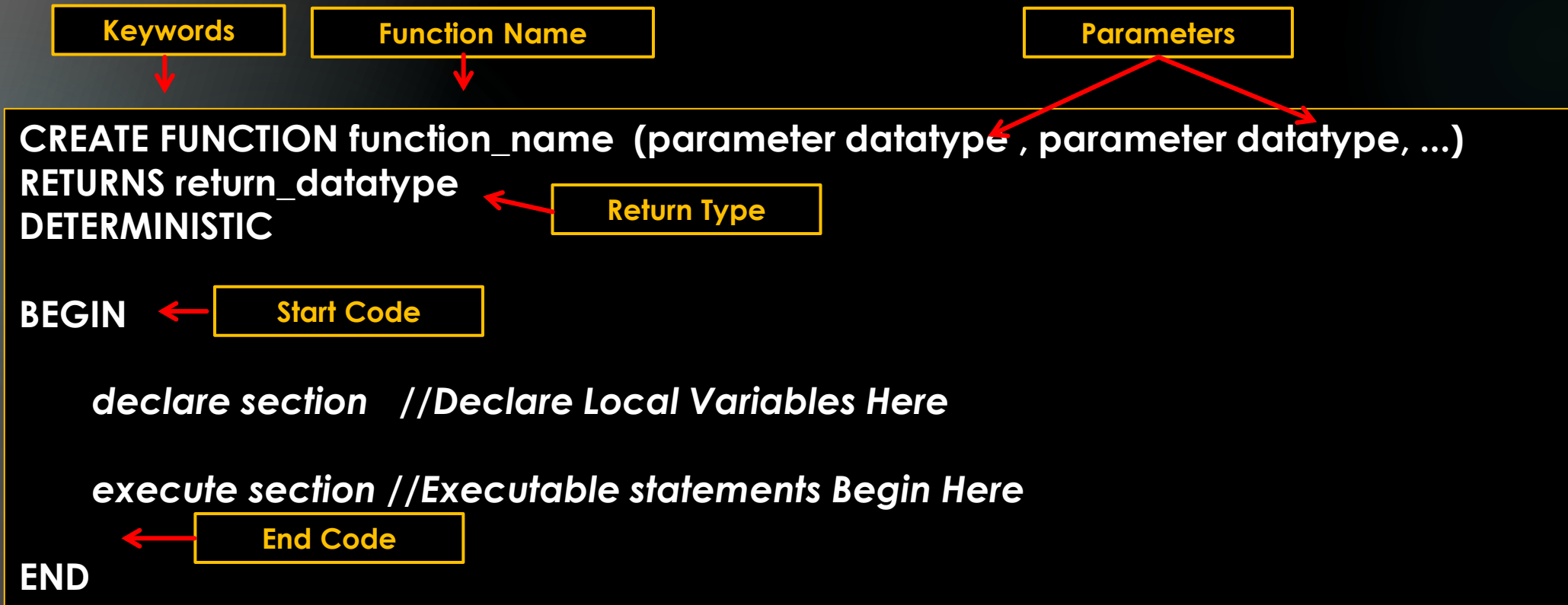
  ▶ **Event**

    ▶ Is executed at a scheduled time.

General

▶ **MySQL supports three types of programming structures.**

Types of Stored Programs		
Type		**Description**
Stored Routines	**Stored Procedure**	Can be called from a SQL statement Can be called from an application that has access to the database.
	Stored Function	Can be called from a SQL statement. Can be considered a user-defined function.
Trigger		Is executed in response to an INSERT, UPDATE, or DELETE statement on a specific table.
Event		Is executed at a scheduled time.

User Defined Functions

▶ A **user-defined function** (UDF) is a way to extend MySQL with a new function that works like a native (built-in) MySQL function.

▶ The syntax for a UDF is:

Keywords	Function Name			Parameters

CREATE FUNCTION function_name (parameter datatype , parameter datatype, ...)
RETURNS return_datatype   ← Return Type
DETERMINISTIC

BEGIN ← Start Code

   declare section   //Declare Local Variables Here

   execute section //Executable statements Begin Here
   ← End Code
END

User-Defined Functions

```sql
C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Su...
File   Edit   Selection   Find   View   Goto   Tools   Project   Preferences   Help

functionTesting.sql

1  DROP DATABASE IF EXISTS functionDEMO;
2  CREATE DATABASE functionDEMO;
3  USE functionDEMO;
4
5  DELIMITER //
6  CREATE FUNCTION addMe(A INT, B INT)
7  RETURNS INT
8  DETERMINISTIC
9
10 BEGIN
11     DECLARE C INT;
12     SET C = A + B;
13     RETURN C;
14
15 END //
16 DELIMITER ;
17
18 SELECT addMe(10,20) AS VALUE;
19

Line 1, Column 37                     Tab Size: 4         SQL
```
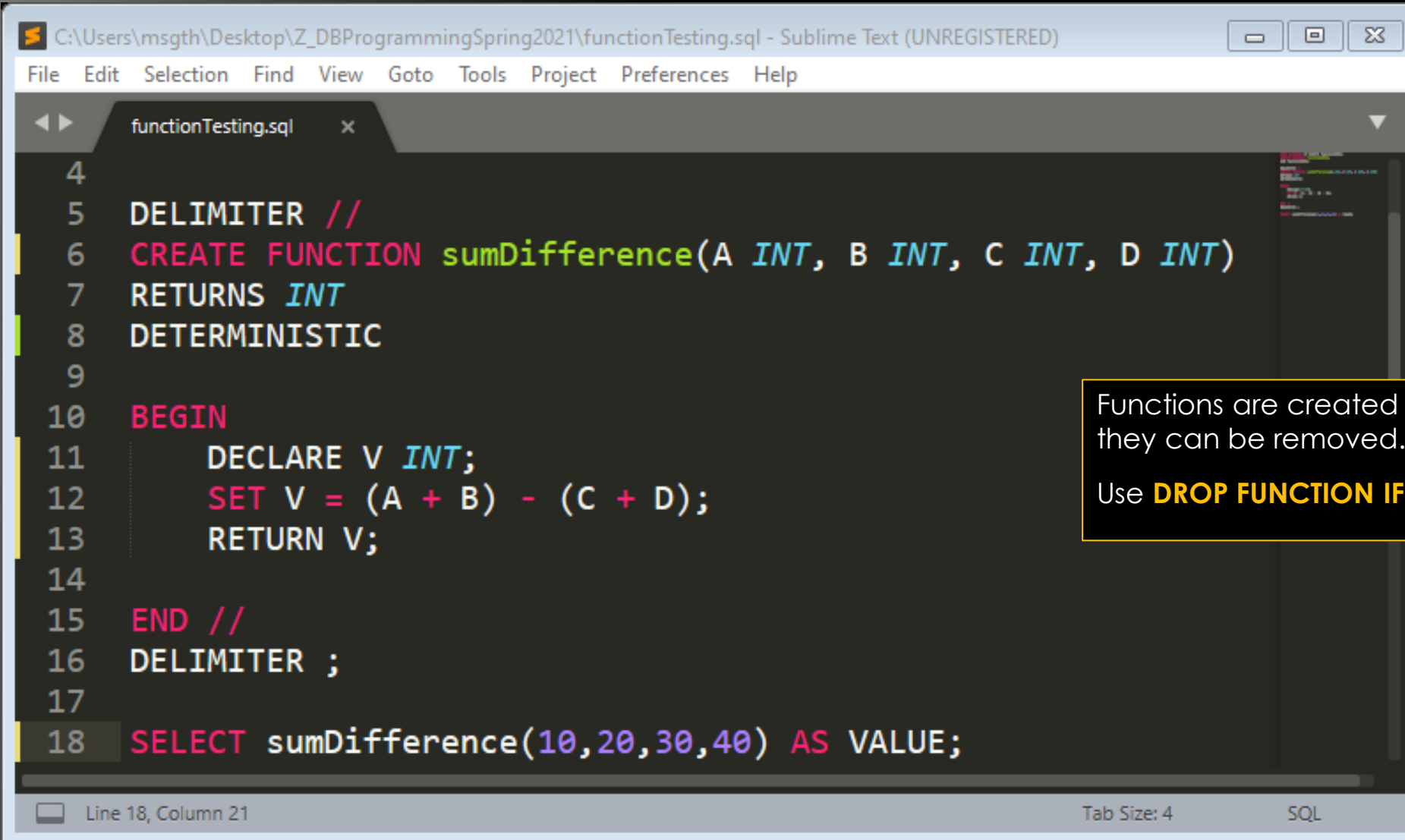
Parameters

Statements

Arguments

```
MySQL 5.7 Com...

+-------+
| Value |
+-------+
|    30 |
+-------+
```

User-Defined Functions

```
C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime Text (UNREGISTERED)

File  Edit  Selection  Find  View  Goto  Tools  Project  Preferences  Help

functionTesting.sql      ×

 4
 5   DELIMITER //
 6   CREATE FUNCTION sumDifference(A INT, B INT, C INT, D INT)
 7   RETURNS INT
 8   DETERMINISTIC
 9
10   BEGIN
11       DECLARE V INT;
12       SET V = (A + B) - (C + D);
13       RETURN V;
14
15   END //
16   DELIMITER ;
17
18   SELECT sumDifference(10,20,30,40) AS VALUE;

Line 18, Column 21                           Tab Size: 4        SQL
```

Functions are created on the server (Globally) and they can be removed.

Use **DROP FUNCTION IF EXISTS**...

Functions and Flow Control

▶ MySQL provides statements that can be used within scripts to add functionality similar to that provided by procedural languages.

▶ **Flow Control** allows the function to create branches in execution based on conditions.

Flow Control	
Keyword	**Description**
IF...THEN ELSEIF...ELSE...END IF	Controls the flow of execution based on a condition.
CASE...WHEN...ELSE...END CASE	

IF...THEN

```
C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime Text (UNREGISTERED)
File  Edit  Selection  Find  View  Goto  Tools  Project  Preferences  Help

functionTesting.sql        ×

1   DROP DATABASE IF EXISTS functionDEMO;
2   CREATE DATABASE functionDEMO;
3   USE functionDEMO;
4
5   DELIMITER //
6   CREATE FUNCTION bigSmallValue(A INT, B INT, C INT)
7   RETURNS VARCHAR(20)
8   DETERMINISTIC
9
10  BEGIN
11      DECLARE S INT;
12      DECLARE V VARCHAR(20);
13      SET S = (A + B + C);
14      IF S > 99 THEN
15          SET V = 'Big Value';
16      END IF;
17      IF S < 99 THEN
18          SET V = 'Small Value';
19      END IF;
20      RETURN V;
21
22  END //
23  DELIMITER ;
24
25  SELECT bigSmallValue(10,20,30) AS MSG;
26

Line 25, Column 39
```

Conditional Statements

```
+-------------+
| MSG         |
+-------------+
| Small Value |
+-------------+
1 row in set (0.08 sec)
```

- The **IF...THEN** statement is used to execute one or more statements depending on a Boolean expressions.
- **IF...THEN** statements can be nested within another **IF...THEN** statement.

```
C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime Text (UNREGISTERED)

File   Edit   Selection   Find   View   Goto   Tools   Project   Preferences   Help

   functionTesting.sql   ×

 1   DROP DATABASE IF EXISTS functionDEMO;
 2   CREATE DATABASE functionDEMO;
 3   USE functionDEMO;
 4
 5   DELIMITER //
 6   CREATE FUNCTION bigSmallValue(A INT, B INT, C INT)
 7   RETURNS VARCHAR(20)
 8   DETERMINISTIC
 9
10   BEGIN
11       DECLARE S INT;
12       DECLARE V VARCHAR(20);
13       SET S = (A + B + C);
14       IF S > 99 THEN
15           SET V = 'Big Value';
16       ELSE
17           SET V = 'Small Value';
18       END IF;
19       RETURN V;
20
21   END //
22   DELIMITER ;
23
24   SELECT bigSmallValue(80,20,30) AS MSG;
25

Line 24, Column 23                                    Tab Size: 4
```

```
+-----------+
| MSG       |
+-----------+
| Big Value |
+-----------+
1 row in set (0.00 sec)
```

- The **IF...THEN ELSE** statement is used to execute one or more statements depending on one or more Boolean expressions.

Nested IF

```
4
5   DELIMITER //
6   CREATE FUNCTION bigSmallValue(A INT, B INT, C INT)
7   RETURNS VARCHAR(20)
8   DETERMINISTIC
9
10  BEGIN
11      DECLARE S INT;
12      DECLARE V VARCHAR(20);
13      SET S = (A + B + C);
14      IF S > 99 THEN
15          SET V = 'Big Value';
16          IF S = 100 THEN
17              SET V = 'Value is 100';
18          END IF;
19      ELSE
20          SET V = 'Small Value';
21      END IF;
22      RETURN V;
23
24  END //
25  DELIMITER ;
26
27  SELECT bigSmallValue(80,20,0) AS MSG;
28
```

Nested IF Statement

- **Nested IF statements** allow the testing of multiple conditions.
- Is considered poor style but sometimes may be expedient.

```
+--------------+
| MSG          |
+--------------+
| Value is 100 |
+--------------+
1 row in set (0.00 sec)
```

C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

functionTesting.sql

Line 27, Column 28   Tab Size: 4

```sql
DELIMITER //
CREATE FUNCTION bigSmallValue(A INT, B INT, C INT)
RETURNS VARCHAR(20)
DETERMINISTIC

BEGIN
    DECLARE S INT;
    DECLARE V VARCHAR(20);
    SET S = (A + B + C);
    IF S > 99 THEN
        SET V = 'Big Value';
    ELSEIF S = 100 THEN
        SET V = 'Value is 100';
    ELSEIF S = 150 THEN
        SET V = 'Value is 150';
    ELSE
        SET V = 'Small Value';
    END IF;
    RETURN V;

END //
DELIMITER ;

SELECT bigSmallValue(80,60,10) AS MSG;
```

C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime Text (UNREGISTERED)

File   Edit   Selection   Find   View   Goto   Tools   Project   Preferences   Help

functionTesting.sql

Line 28, Column 29   Tab Size: 4

ELSEIF Statement

- **IF… THEN…ELSEIF** tests multiple conditions and **executes the first** true condition.
- This condition is the **MOST** true but it is not the first true statement

```sql
ELSEIF S = 150 THEN
        SET V = 'Value is 150';
```

```
+-----------+
| MSG       |
+-----------+
| Big Value |
+-----------+
1 row in set (0.00 sec)
```

CASE...END CASE

```
C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime Text (UNREGISTERED)
File   Edit   Selection   Find   View   Goto   Tools   Project   Preferences   Help

   functionTesting.sql    ×

 4
 5   DELIMITER //
 6   CREATE FUNCTION bigSmallValue(A INT, B INT, C INT)
 7   RETURNS VARCHAR(20)
 8   DETERMINISTIC
 9
10   BEGIN
11       DECLARE testMe INT;
12       DECLARE V VARCHAR(20);
13       SET testMe = (A + B + C);
14       CASE testMe
15           WHEN 100 THEN
16               SET V = '100';
17           WHEN 150 THEN
18               SET V = '150';
19           ELSE
20               SET V = '???';
21       END CASE;
22       RETURN V;
23
24   END //
25   DELIMITER ;
26
27   SELECT bigSmallValue(80,60,10) AS MSG;
28

Line 18, Column 24                          Tab Size: 4         SQL
```

CASE Statement

```
+------+
| MSG  |
+------+
| 150  |
+------+
1 row in set (0.00 sec)
```

- **CASE statement structure** allow the testing of multiple conditions using pattern matching.
- The first true 'CASE' is executed.

Functions and Loops

▶ **MySQL provides <u>three types</u> of loops.**

Loops	
Keyword	**Description**
WHILE…DO..END WHILE	Repeats statements while a condition is true.
LOOP…LEAVE…END LOOP	
REPEAT…UNTIL…END REPEAT	

WHILE...DO...END WHILE Loop

```
C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - ...
File  Edit  Selection  Find  View  Goto  Tools  Project  Preferences  Help

functionTesting.sql      ×

 4
 5   DELIMITER //
 6   CREATE FUNCTION incrementBy10(A INT)
 7   RETURNS VARCHAR(20)
 8   DETERMINISTIC
 9
10   BEGIN
11       DECLARE I INT;
12       DECLARE V VARCHAR(20);
13       SET I = 0;
14       SET V = 0;
15       WHILE I < A DO
16           SET V = V + 10;
17           SET I = I + 1;
18       END WHILE;
19       RETURN V;
20
21   END //
22   DELIMITER ;
23
24   SELECT incrementBy10(4) AS MSG;
25

Line 20, Column 1                                    Tab Size: 4
```

WHILE DO Loop

```
+------+
| MSG  |
+------+
| 40   |
+------+
1 row in set (0.00 sec)
```

- The **WHILE DO** loop executes as long as the loop continuation condition is true.

LOOP…LEAVE…END LOOP Loop

```sql
4
5   DELIMITER //
6   CREATE FUNCTION incrementBy10(A INT)
7   RETURNS VARCHAR(20)
8   DETERMINISTIC
9
10  BEGIN
11      DECLARE I INT;
12      DECLARE V VARCHAR(20);
13      SET I = 0;
14      SET V = 0;
15      myLoop: LOOP
16          SET V = V + 10;
17          SET I = I + 1;
18      IF I = A THEN
19          LEAVE myLoop;
20      END IF;
21      END LOOP myLoop;
22      RETURN V;
23
24  END //
25  DELIMITER ;
26
27  SELECT incrementBy10(4) AS MSG;
```

LOOP END LOOP Loop

```
+------+
| MSG  |
+------+
| 40   |
+------+
1 row in set (0.00 sec)
```

- The **LOOP END LOOP** loop executes if the loop continuation condition is true.
- The loop continuation condition is specified by use of a conditional statement as uses the **LEAVE** keyword.

REPEAT…UNTIL…END REPEAT Loop

```sql
DELIMITER //
CREATE FUNCTION incrementBy10(A INT)
RETURNS VARCHAR(20)
DETERMINISTIC

BEGIN
    DECLARE I INT;
    DECLARE V VARCHAR(20);
    SET I = 0;
    SET V = 0;
    REPEAT
        SET V = V + 10;
        SET I = I + 1;
    UNTIL I = A
    END REPEAT;
    RETURN V;

END //
DELIMITER ;

SELECT incrementBy10(4) AS MSG;
```

REPEAT LOOP Loop

```
+------+
| MSG  |
+------+
| 40   |
+------+
1 row in set (0.00 sec)
```

- The **REPEAT…UNTIL LOOP** loop executes if the loop continuation condition is true.
- The loop continuation condition is specified by use of the UNTIL keyword.

UDF inside UDF

```sql
DROP DATABASE IF EXISTS functionDEMO;
CREATE DATABASE functionDEMO;
USE functionDEMO;


DELIMITER //
CREATE FUNCTION plusDifference(A DECIMAL(10,2), B INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC

BEGIN
    DECLARE S DECIMAL(10,2);
    DECLARE T DECIMAL(10,2);
    DECLARE U DECIMAL(10,2);

    SET S = addMe(A,B);
    SET T = subtractMe(A,B);
    SET U = S + T;
    RETURN U;

END //
DELIMITER ;

SELECT plusDifference(10,20) AS MSG;
```

```sql
DELIMITER //
CREATE FUNCTION addMe(A INT, B INT)
RETURNS INT
DETERMINISTIC

BEGIN
    DECLARE C INT;
    SET C = A + B;
    RETURN C;

END //
DELIMITER ;
```
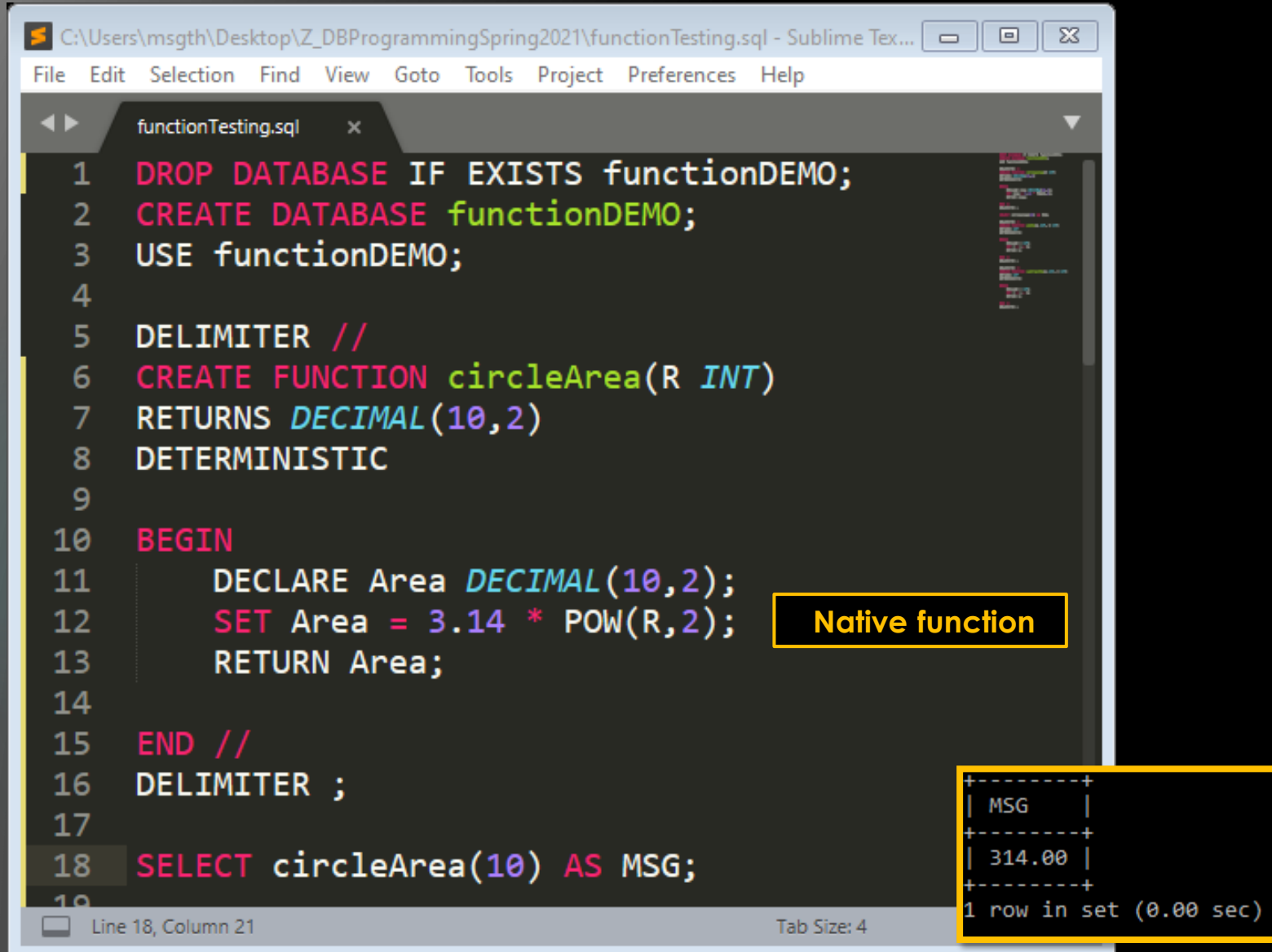
```sql
DELIMITER //
CREATE FUNCTION subtractMe(A INT, B INT)
RETURNS INT
DETERMINISTIC

BEGIN
    DECLARE C INT;
    SET C = A - B;
    RETURN C;

END //
DELIMITER ;
```

UDF and Native Function

```
C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime Tex...
File   Edit   Selection   Find   View   Goto   Tools   Project   Preferences   Help

functionTesting.sql    ×

 1   DROP DATABASE IF EXISTS functionDEMO;
 2   CREATE DATABASE functionDEMO;
 3   USE functionDEMO;
 4
 5   DELIMITER //
 6   CREATE FUNCTION circleArea(R INT)
 7   RETURNS DECIMAL(10,2)
 8   DETERMINISTIC
 9
10   BEGIN
11       DECLARE Area DECIMAL(10,2);
12       SET Area = 3.14 * POW(R,2);        Native function
13       RETURN Area;
14
15   END //
16   DELIMITER ;
17
18   SELECT circleArea(10) AS MSG;
19

Line 18, Column 21                          Tab Size: 4
```
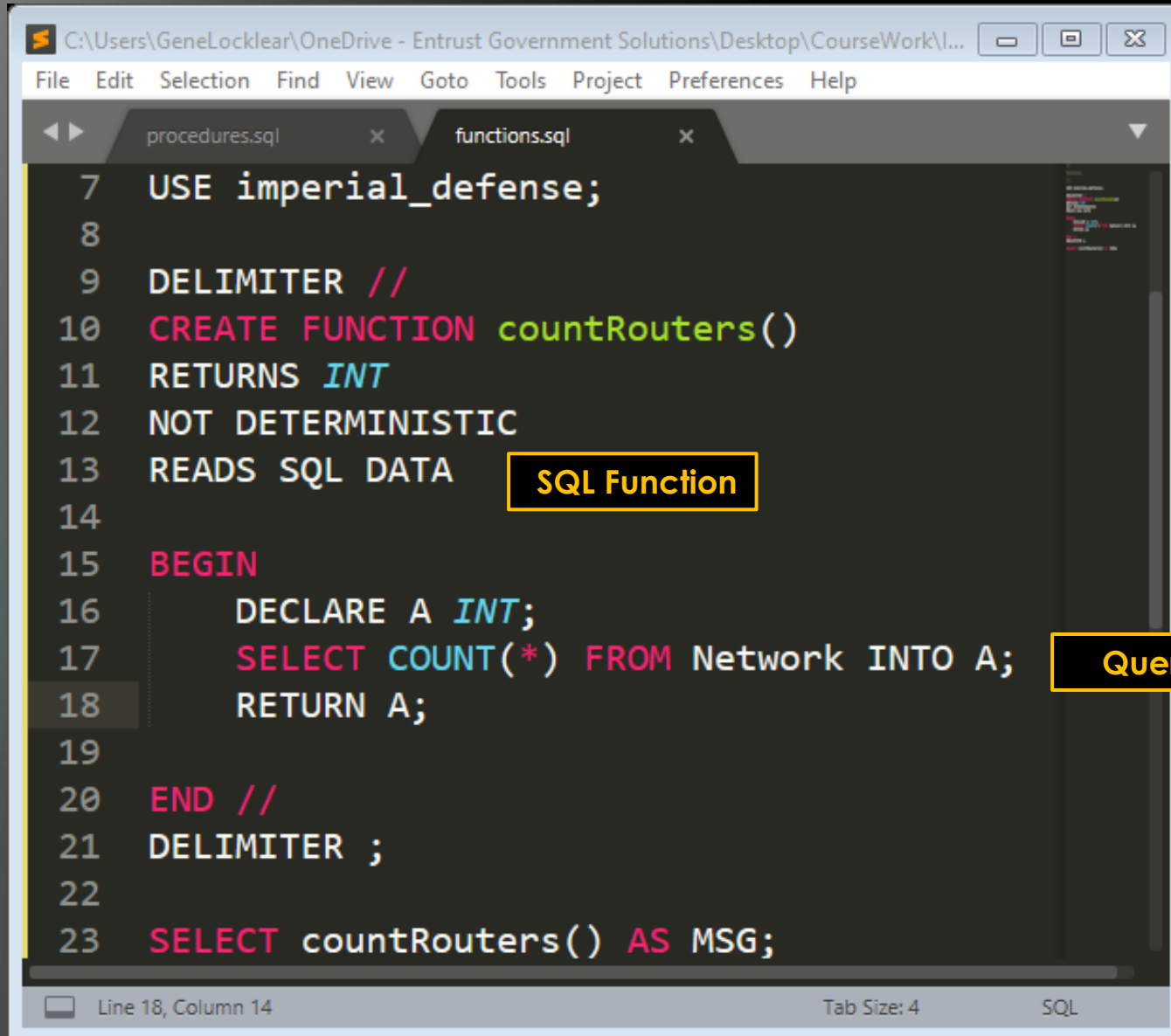
```
+----------+
| MSG      |
+----------+
| 314.00   |
+----------+
1 row in set (0.00 sec)
```
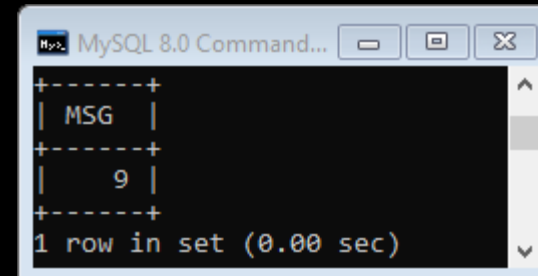
SQL and User-Defined Functions

```sql
7   USE imperial_defense;
8
9   DELIMITER //
10  CREATE FUNCTION countRouters()
11  RETURNS INT
12  NOT DETERMINISTIC
13  READS SQL DATA
14
15  BEGIN
16      DECLARE A INT;
17      SELECT COUNT(*) FROM Network INTO A;
18      RETURN A;
19
20  END //
21  DELIMITER ;
22
23  SELECT countRouters() AS MSG;
```

SQL Function

Query
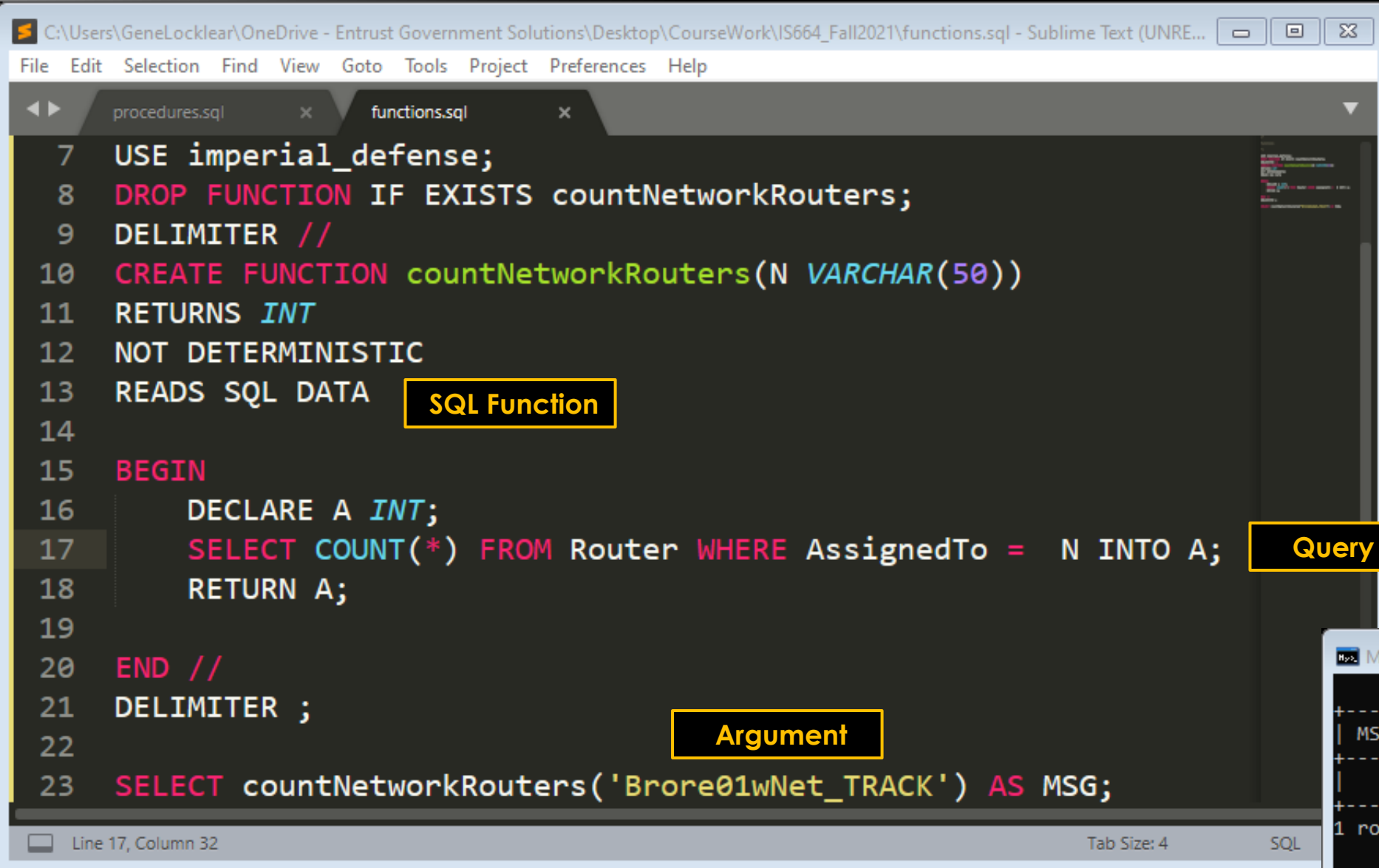
```
+-------+
| MSG   |
+-------+
|     9 |
+-------+
1 row in set (0.00 sec)
```

SQL and User-Defined Functions

C:\Users\GeneLocklear\OneDrive - Entrust Government Solutions\Desktop\CourseWork\IS664_Fall2021\functions.sql - Sublime Text (UNRE...

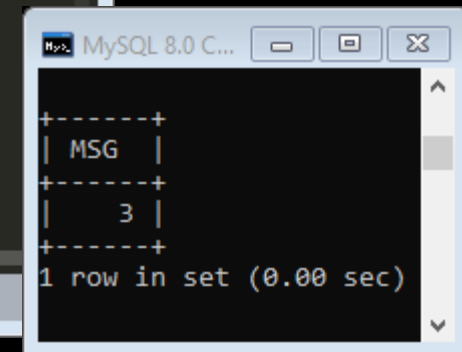File   Edit   Selection   Find   View   Goto   Tools   Project   Preferences   Help

procedures.sql   ✕   functions.sql   ✕

```
 7   USE imperial_defense;
 8   DROP FUNCTION IF EXISTS countNetworkRouters;
 9   DELIMITER //
10   CREATE FUNCTION countNetworkRouters(N VARCHAR(50))
11   RETURNS INT
12   NOT DETERMINISTIC
13   READS SQL DATA              SQL Function
14
15   BEGIN
16       DECLARE A INT;
17       SELECT COUNT(*) FROM Router WHERE AssignedTo =  N INTO A;    Query
18       RETURN A;
19
20   END //
21   DELIMITER ;
22                              Argument
23   SELECT countNetworkRouters('Brore01wNet_TRACK') AS MSG;
```

Line 17, Column 32   Tab Size: 4   SQL

MySQL 8.0 C...

```
+------+
| MSG  |
+------+
|    3 |
+------+
1 row in set (0.00 sec)
```

Practical Exercise

▶ **Create a single .sql script which accomplishes the following task.**

1. Create the function **displayScriptAuthor** which displays your name.
   1. Call the function displayScriptAuthor.

2. Create the function **average5** which accepts 5 integer parameters and returns their average.
   1. Call the function with the parameters 1,2,3,4,5

3. Create the function **varianceA** which accepts a single numeric parameter (A) and calculates the variance of the numbers 1 to A
   1. Call the function with the parameter 5

4. Create the function **sigmaA** which accepts a single numeric parameter (A) and calculates the standard deviation of the numbers 1 to A
   1. Call the function with the parameter 5