IS664 Database Programming
Fall 2021

Stored Functions

LECTURE 4: 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.

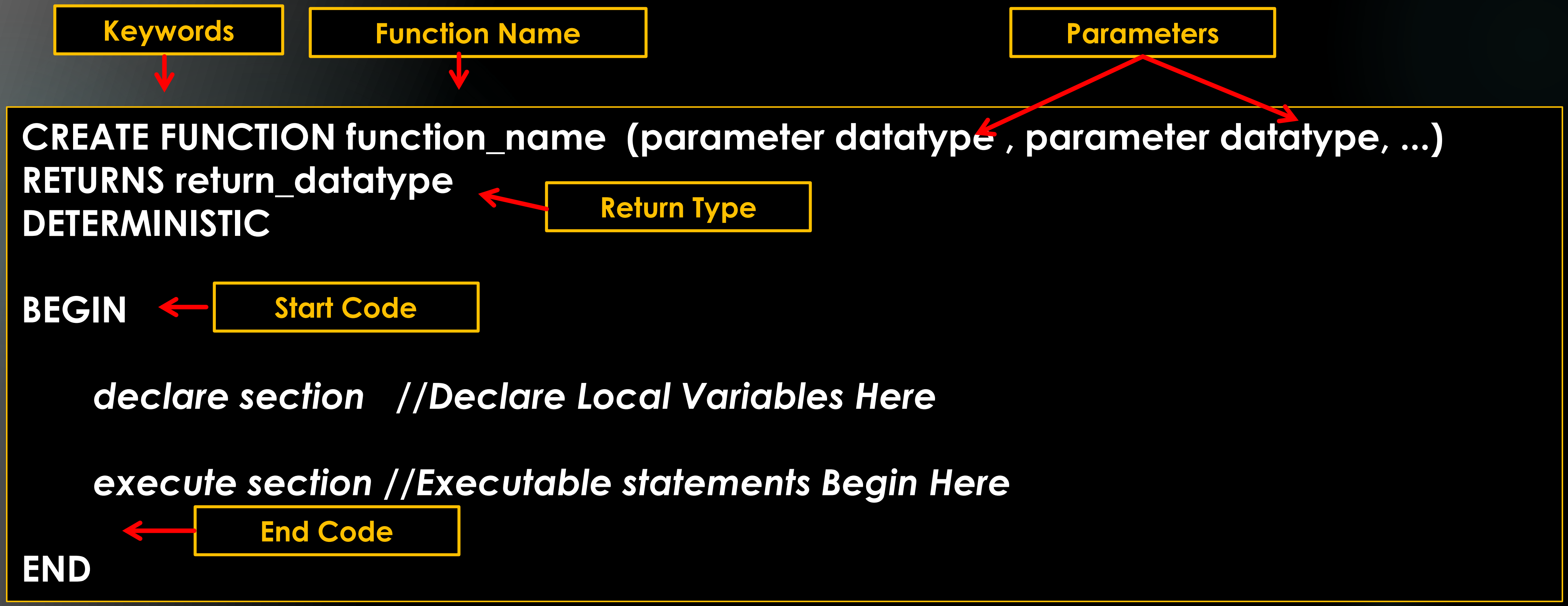
General

- **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.

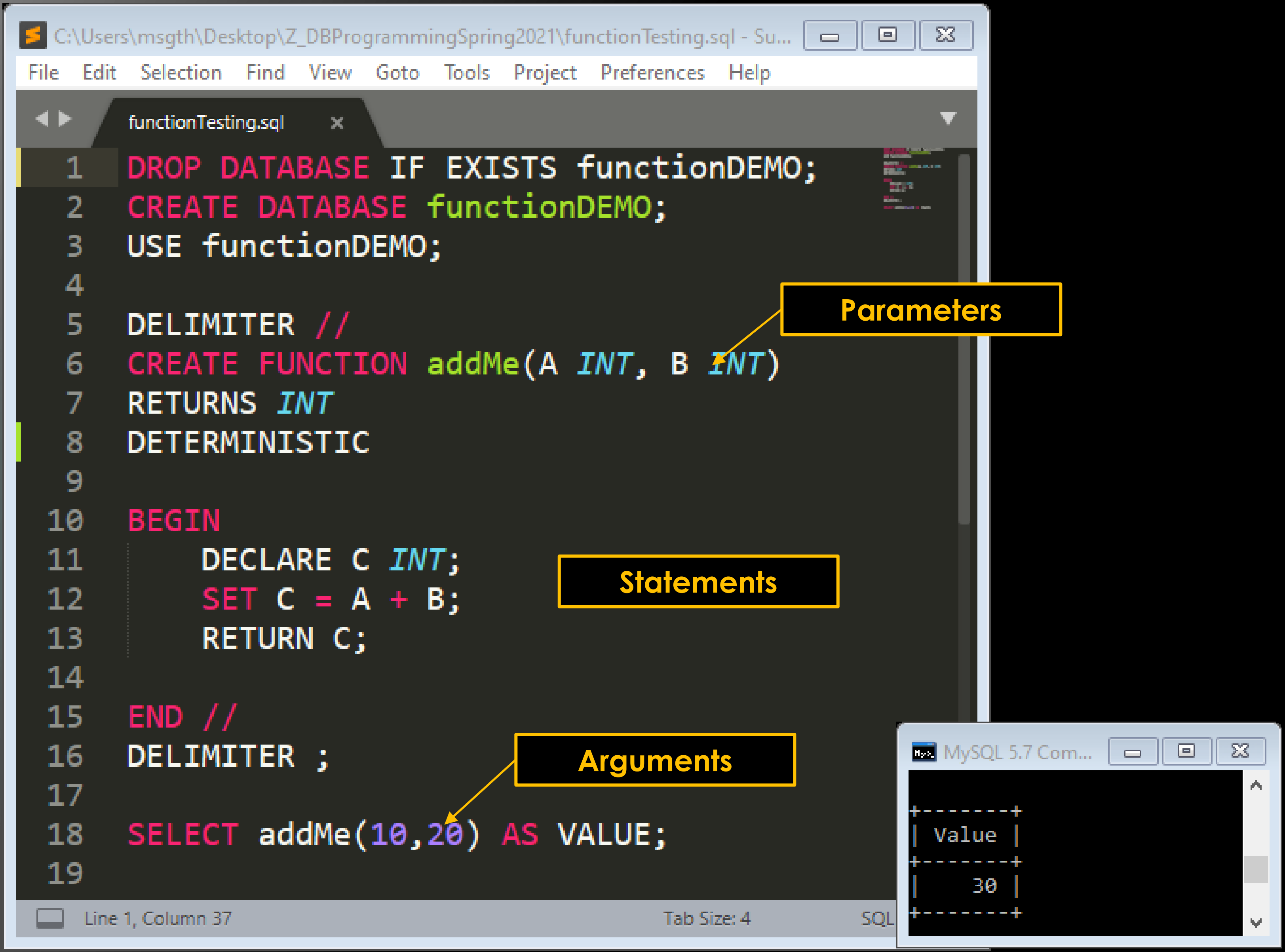
User Defined Functions

- 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:



```
CREATE FUNCTION function_name  (parameter datatype , parameter datatype, ...)
RETURNS return_datatype
DETERMINISTIC

BEGIN

    declare section   //Declare Local Variables Here

    execute section //Executable statements Begin Here

END
```

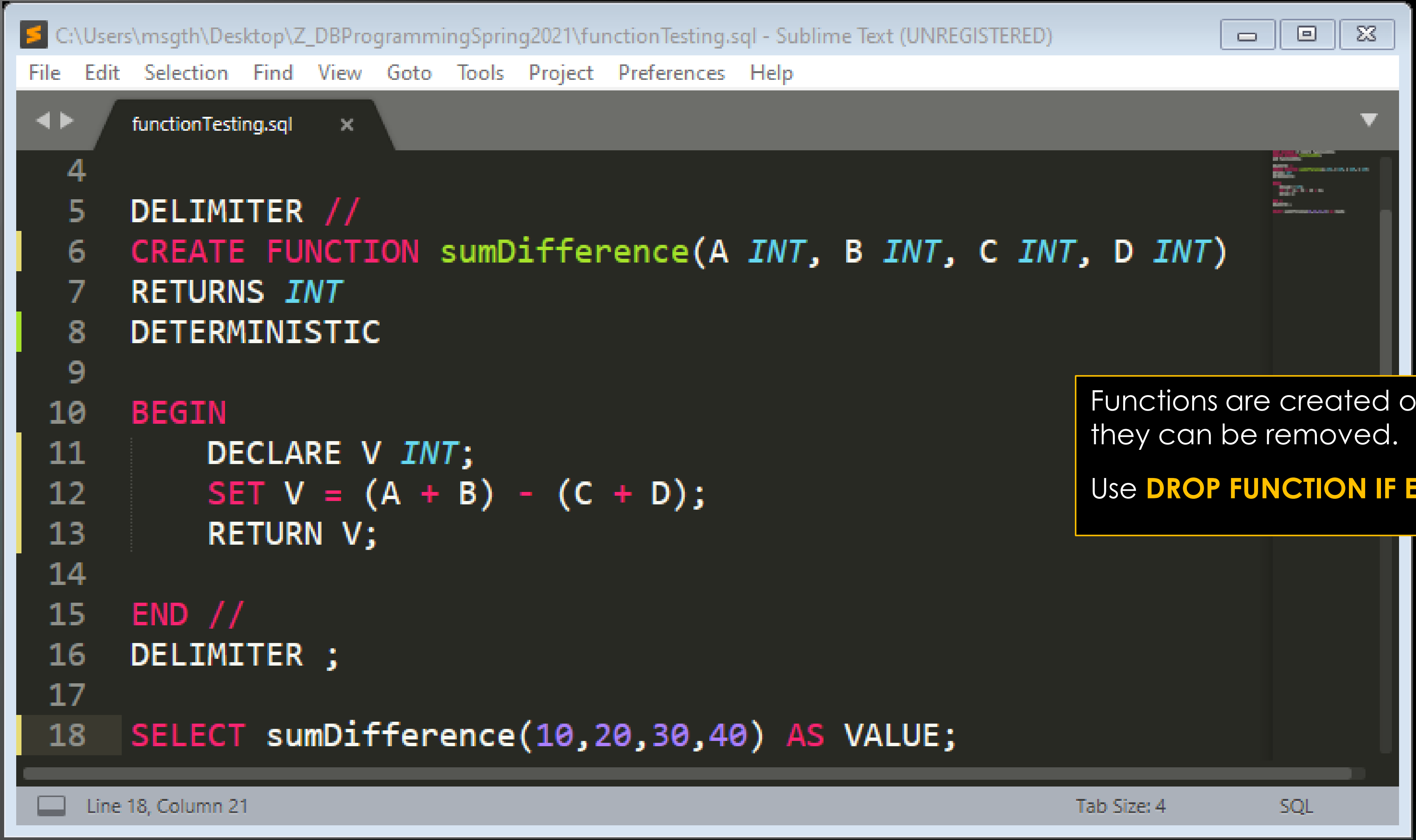
User-Defined Functions



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

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

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

END //
DELIMITER ;

SELECT addMe(10,20) AS VALUE;

```

```
+-------+
| 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
Line 18, Column 21
Tab Size: 4
SQL

```sql

DELIMITER //
CREATE FUNCTION sumDifference(A INT, B INT, C INT, D INT)
RETURNS INT
DETERMINISTIC

BEGIN
    DECLARE V INT;
    SET V = (A + B) - (C + D);
    RETURN V;

END //
DELIMITER ;

SELECT sumDifference(10,20,30,40) AS VALUE;
```

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

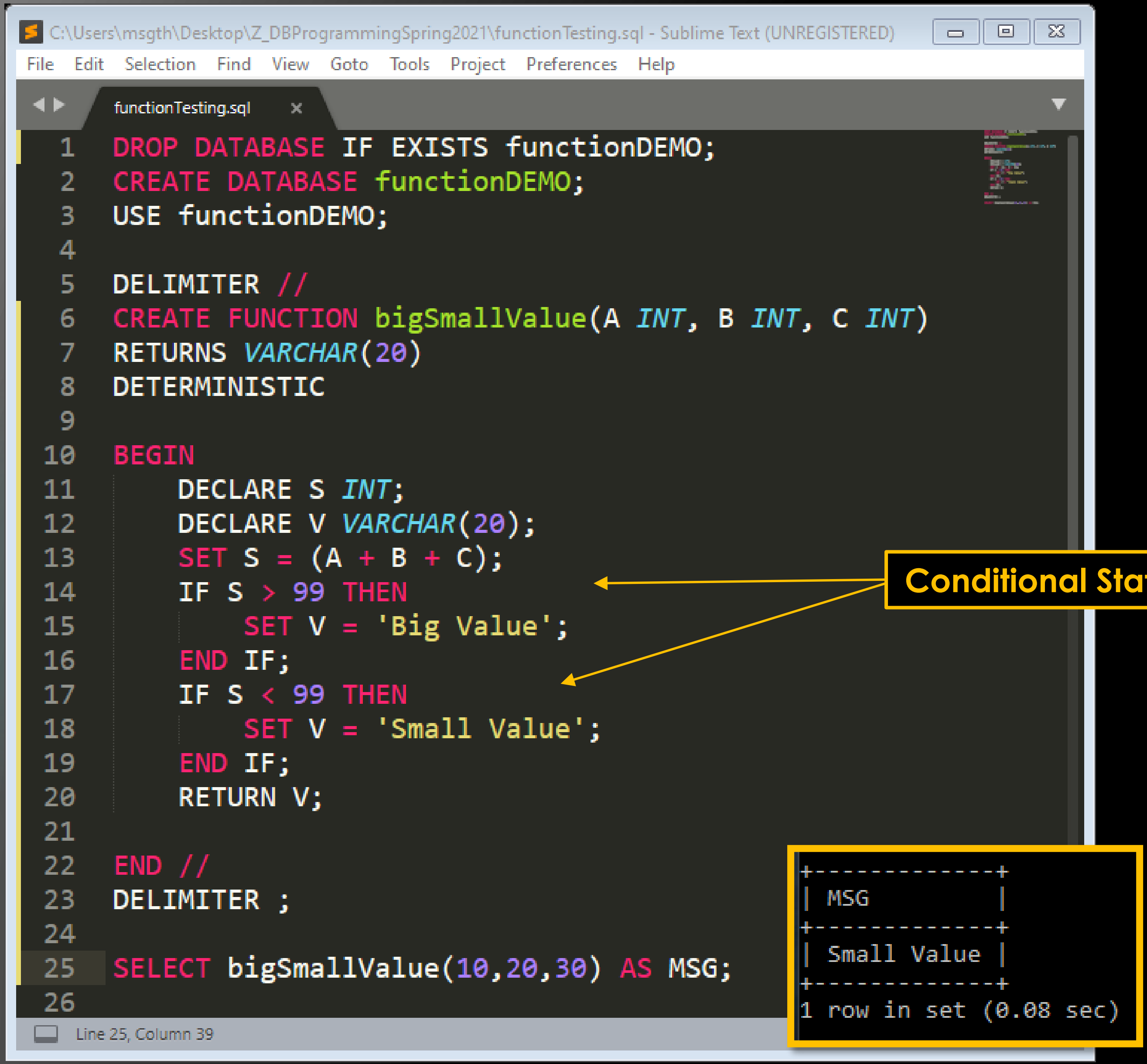
Use **DROP FUNCTION IF EXISTS**...

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
Line 25, Column 39

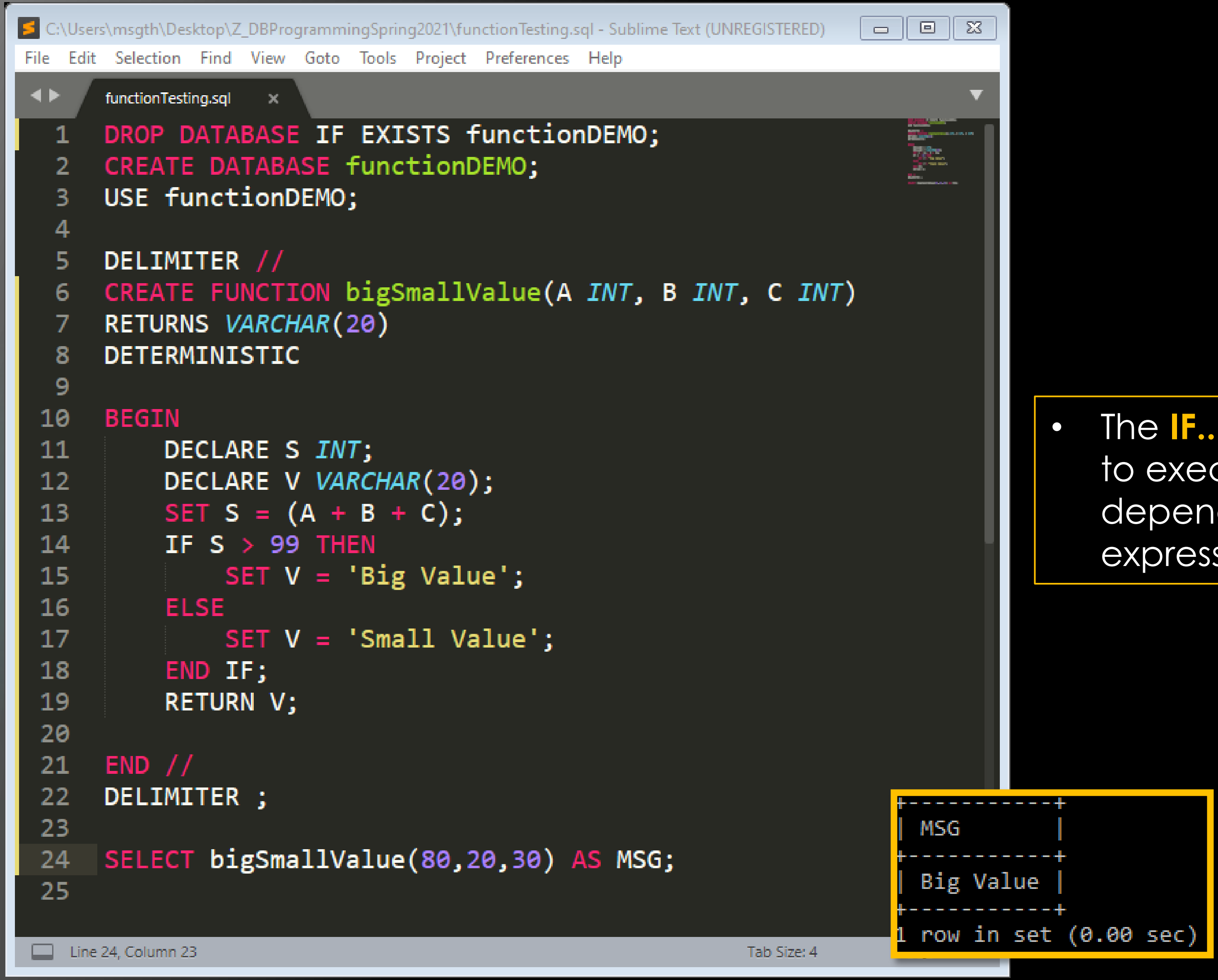
```sql
DROP DATABASE IF EXISTS functionDEMO;
CREATE DATABASE functionDEMO;
USE functionDEMO;

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';
    END IF;
    IF S < 99 THEN
        SET V = 'Small Value';
    END IF;
    RETURN V;

END //
DELIMITER ;

SELECT bigSmallValue(10,20,30) AS MSG;
```

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

Conditional Statements

- 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.

IF…THEN ELSE

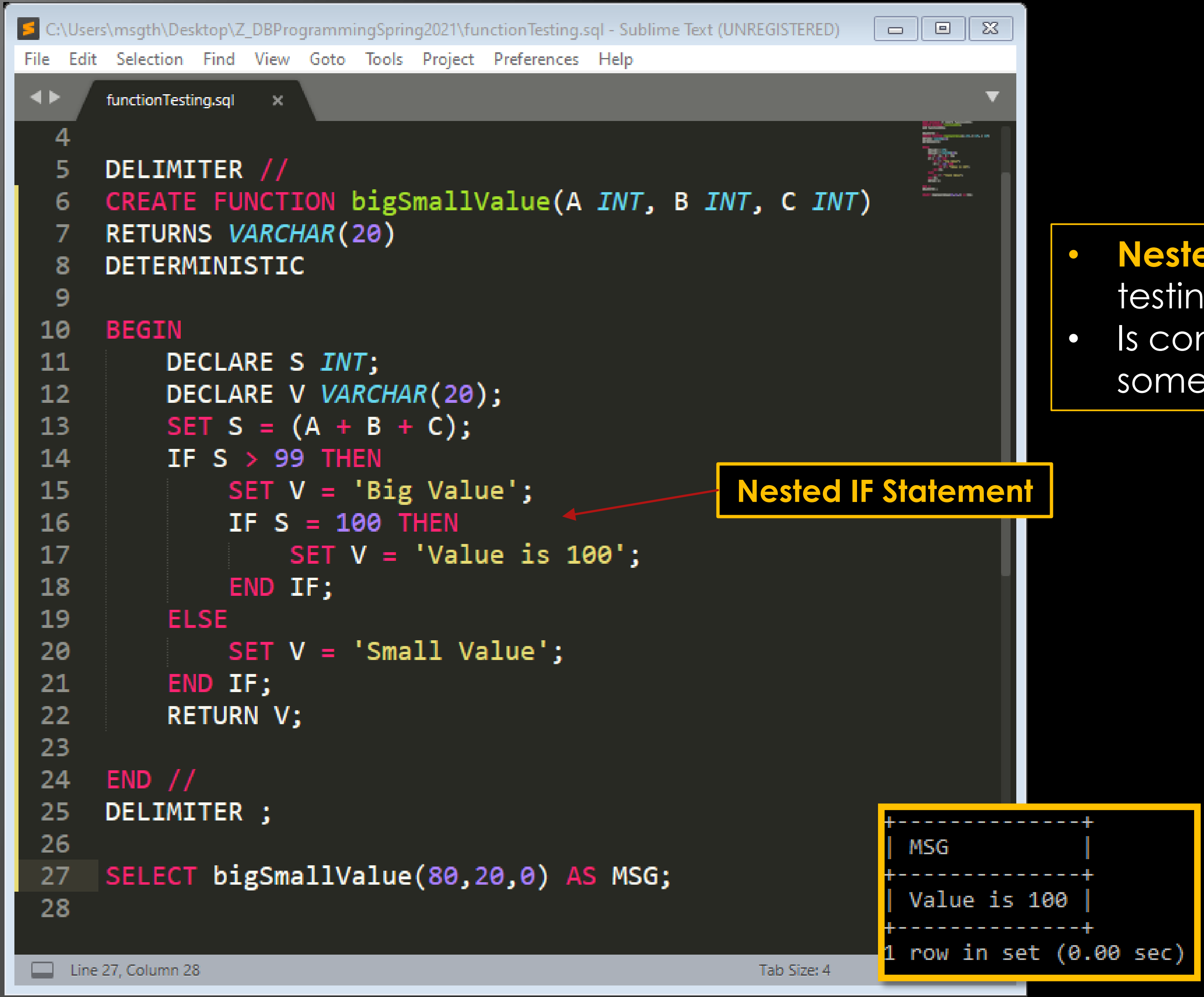

C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
functionTesting.sql
Line 24, Column 23
Tab Size: 4

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

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';
    ELSE
        SET V = 'Small Value';
    END IF;
    RETURN V;

END //
DELIMITER ;

SELECT bigSmallValue(80,20,30) AS MSG;
```

```
+-----------+
| 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



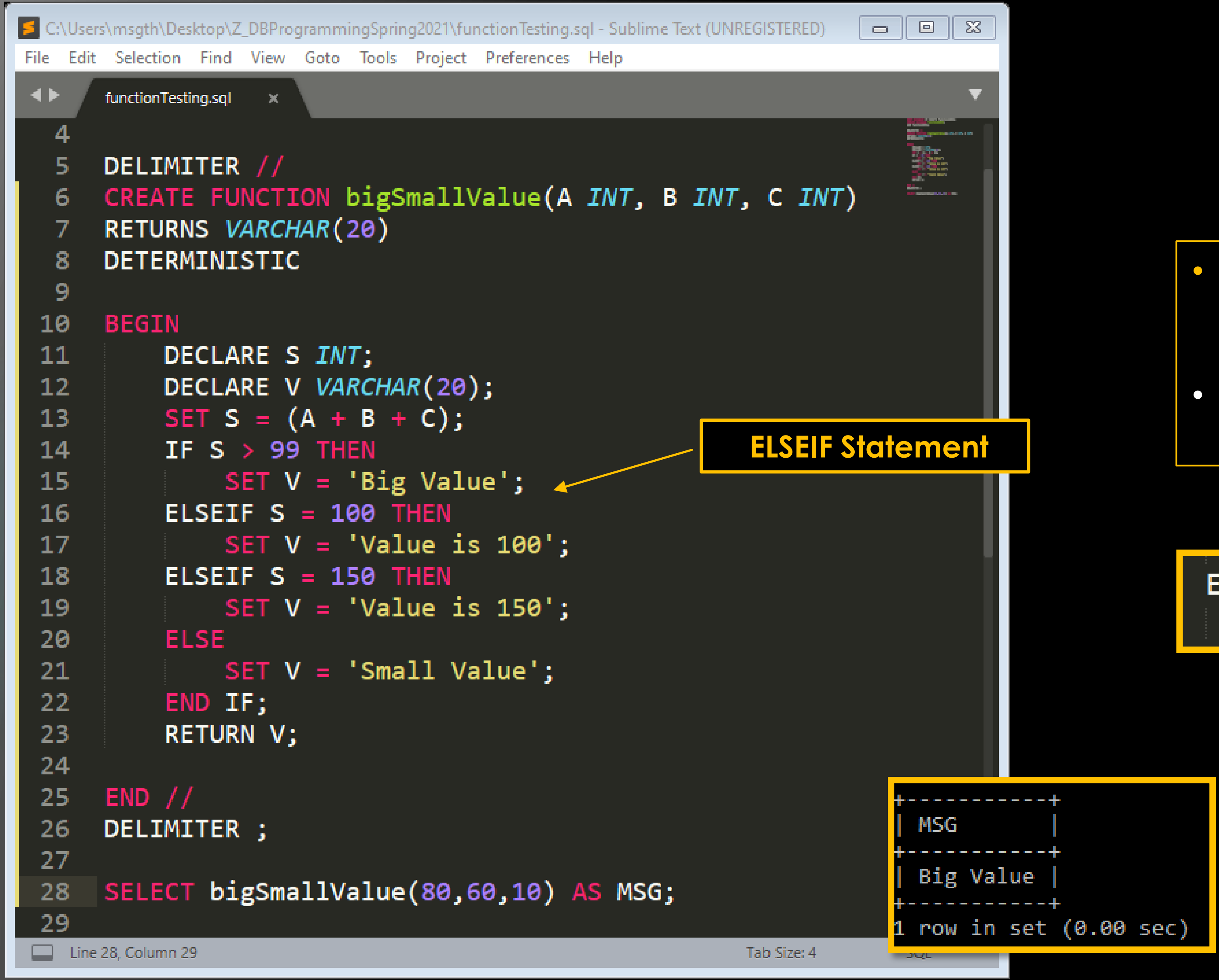
```
DELIMITER //
CREATE FUNCTION bigSmallValue(A INT, B INT, C INT)
RETURNS VARCHAR(20)
DETERMINISTIC

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

END //
DELIMITER ;

SELECT bigSmallValue(80,20,0) AS MSG;
```

Nested IF Statement

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

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

IF…THEN…ELSEIF ELSE

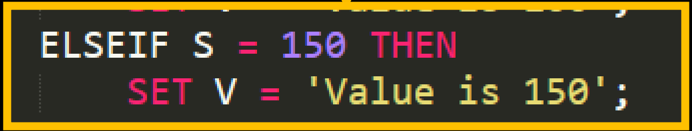


```sql

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;

```

ELSEIF Statement

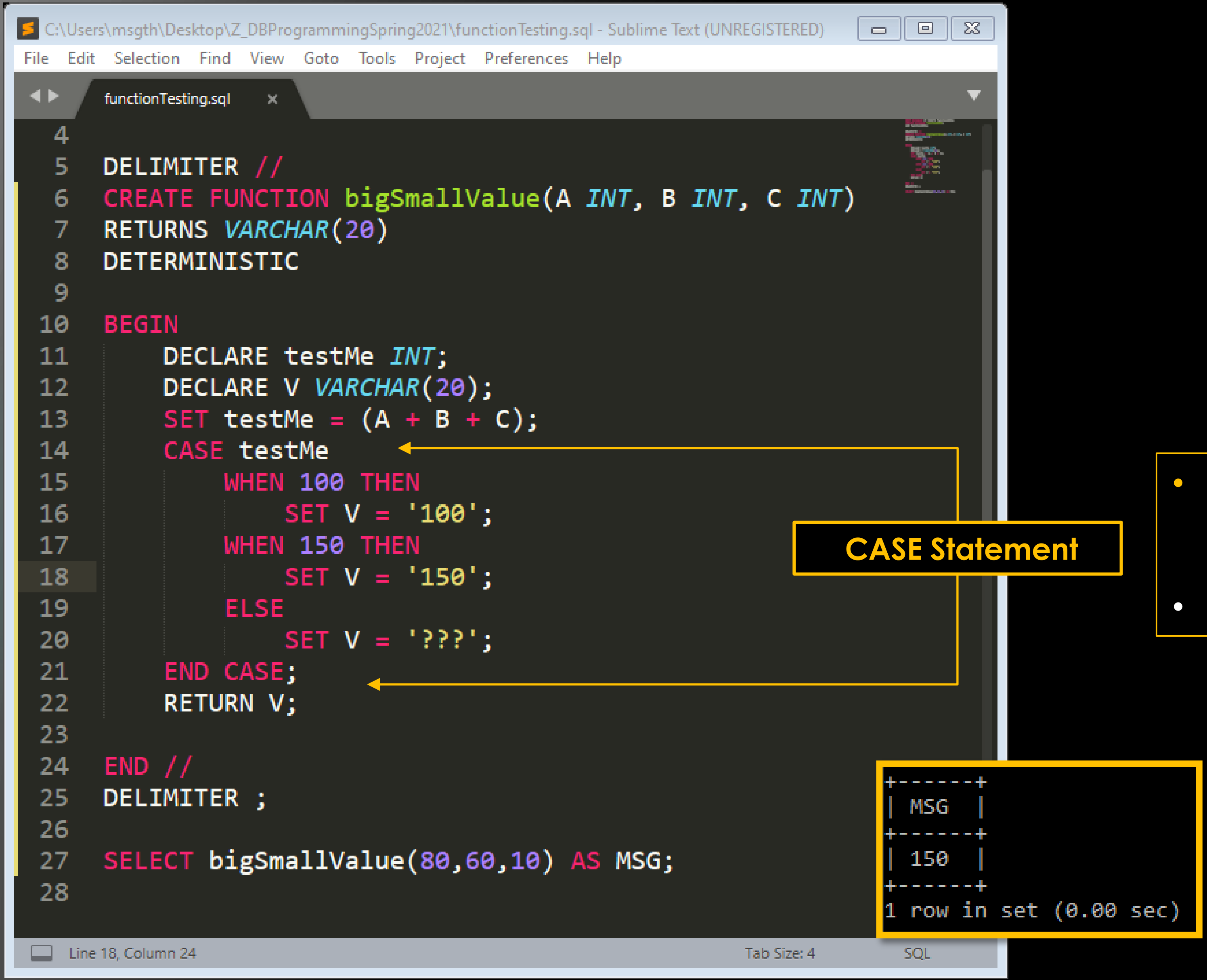
```
+-----------+
| MSG       |
+-----------+
| Big Value |
+-----------+
1 row in set (0.00 sec)
```

- **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';
```

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
Line 18, Column 24
Tab Size: 4
SQL

```

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

BEGIN
    DECLARE testMe INT;
    DECLARE V VARCHAR(20);
    SET testMe = (A + B + C);
    CASE testMe
        WHEN 100 THEN
            SET V = '100';
        WHEN 150 THEN
            SET V = '150';
        ELSE
            SET V = '???';
    END CASE;
    RETURN V;

END //
DELIMITER ;

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

```

CASE Statement

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

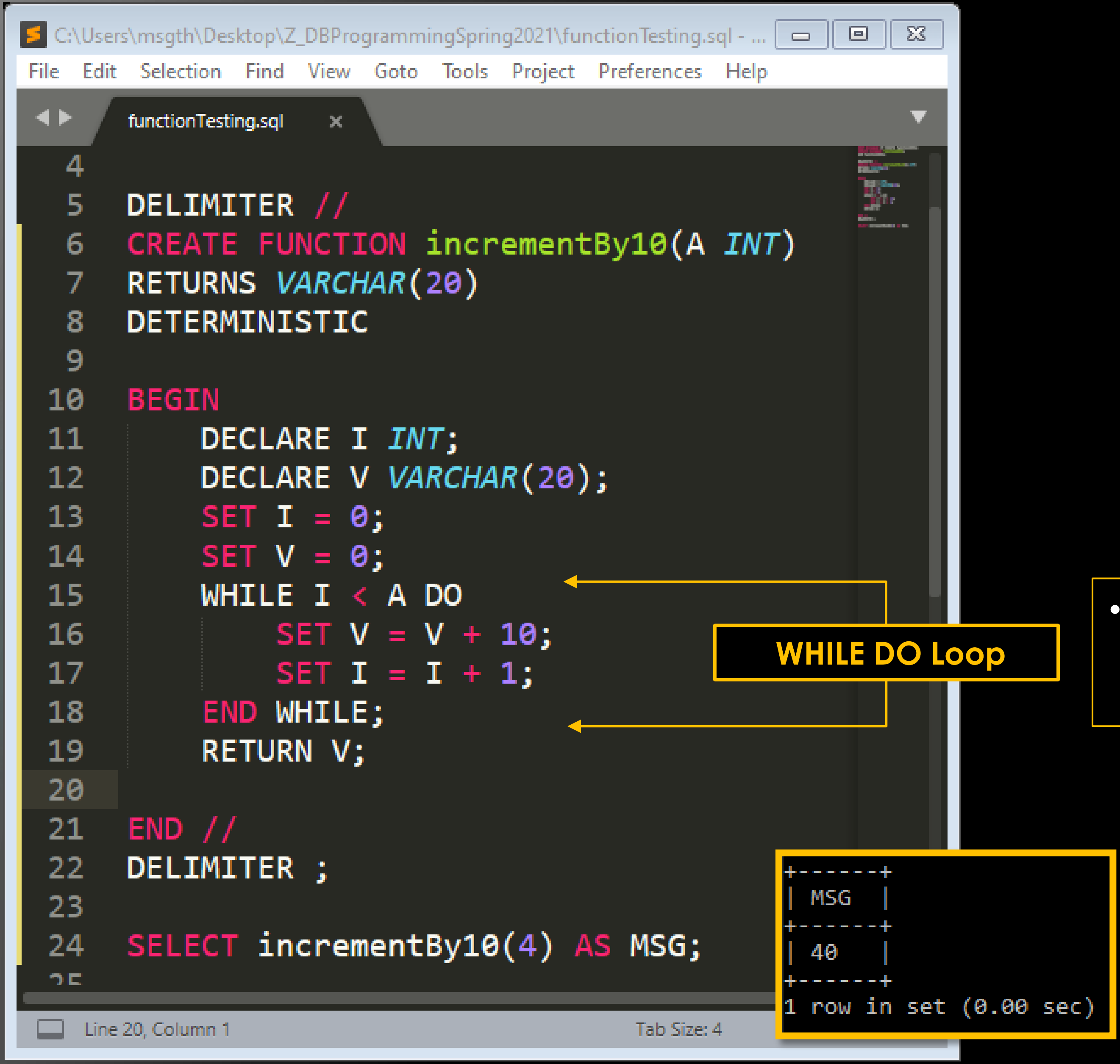
- **CASE statement structure** allow the testing of multiple conditions using pattern matching.
- The first true ‘CASE’ is executed.

Functions and Loops

- **MySQL provides three types 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

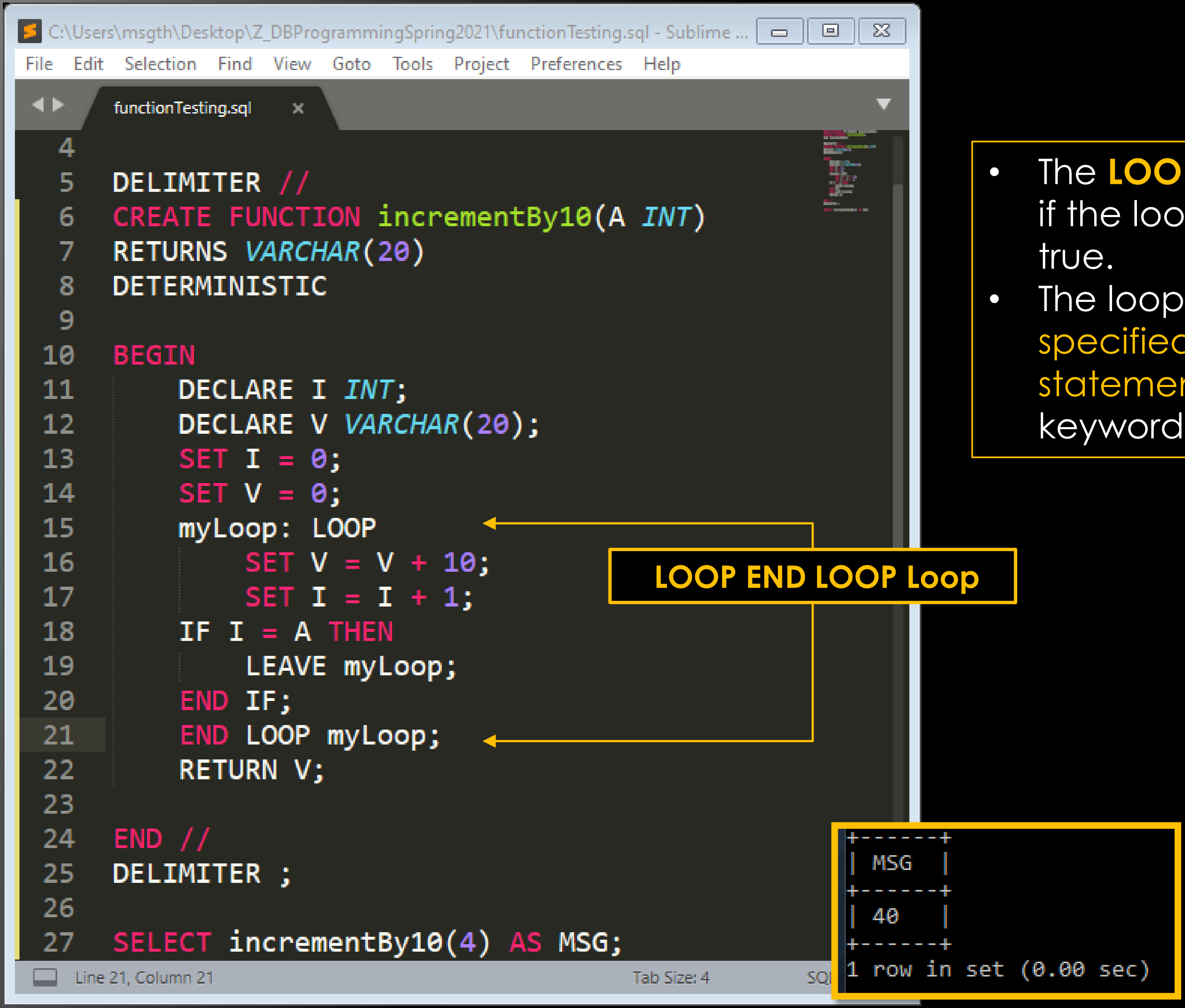

C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - ...
File Edit Selection Find View Goto Tools Project Preferences Help
functionTesting.sql
Line 20, Column 1
Tab Size: 4
WHILE DO Loop

```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;
    WHILE I < A DO
        SET V = V + 10;
        SET I = I + 1;
    END WHILE;
    RETURN V;

END //
DELIMITER ;

SELECT incrementBy10(4) AS MSG;
```

```
+-------+
| 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


C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime ...
File Edit Selection Find View Goto Tools Project Preferences Help
functionTesting.sql
Line 21, Column 21
Tab Size: 4
SQL

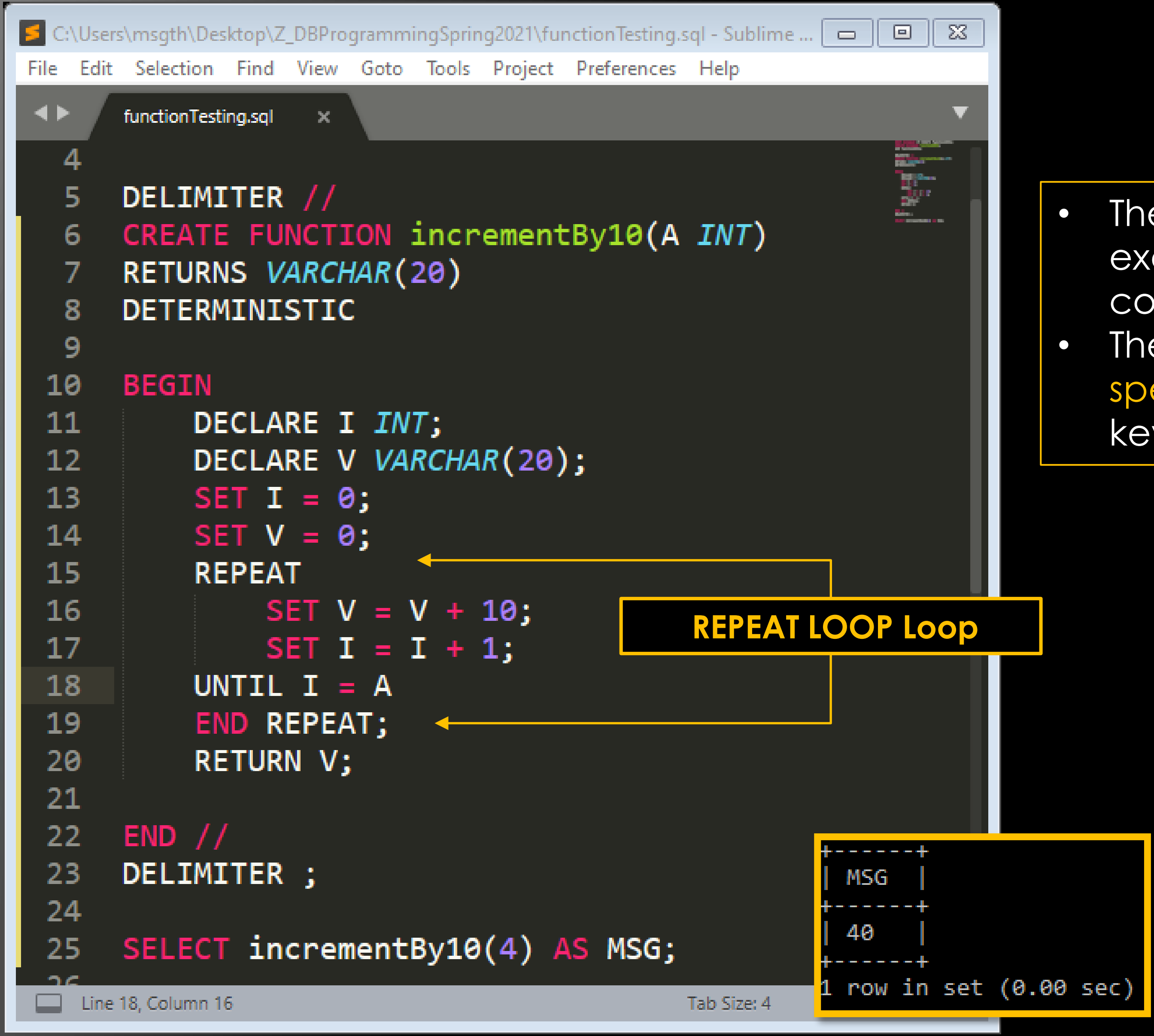
```sql

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

BEGIN
    DECLARE I INT;
    DECLARE V VARCHAR(20);
    SET I = 0;
    SET V = 0;
    myLoop: LOOP
        SET V = V + 10;
        SET I = I + 1;
    IF I = A THEN
        LEAVE myLoop;
    END IF;
    END LOOP myLoop;
    RETURN V;

END //
DELIMITER ;

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


C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime ...
File Edit Selection Find View Goto Tools Project Preferences Help
functionTesting.sql
Line 18, Column 16
Tab Size: 4

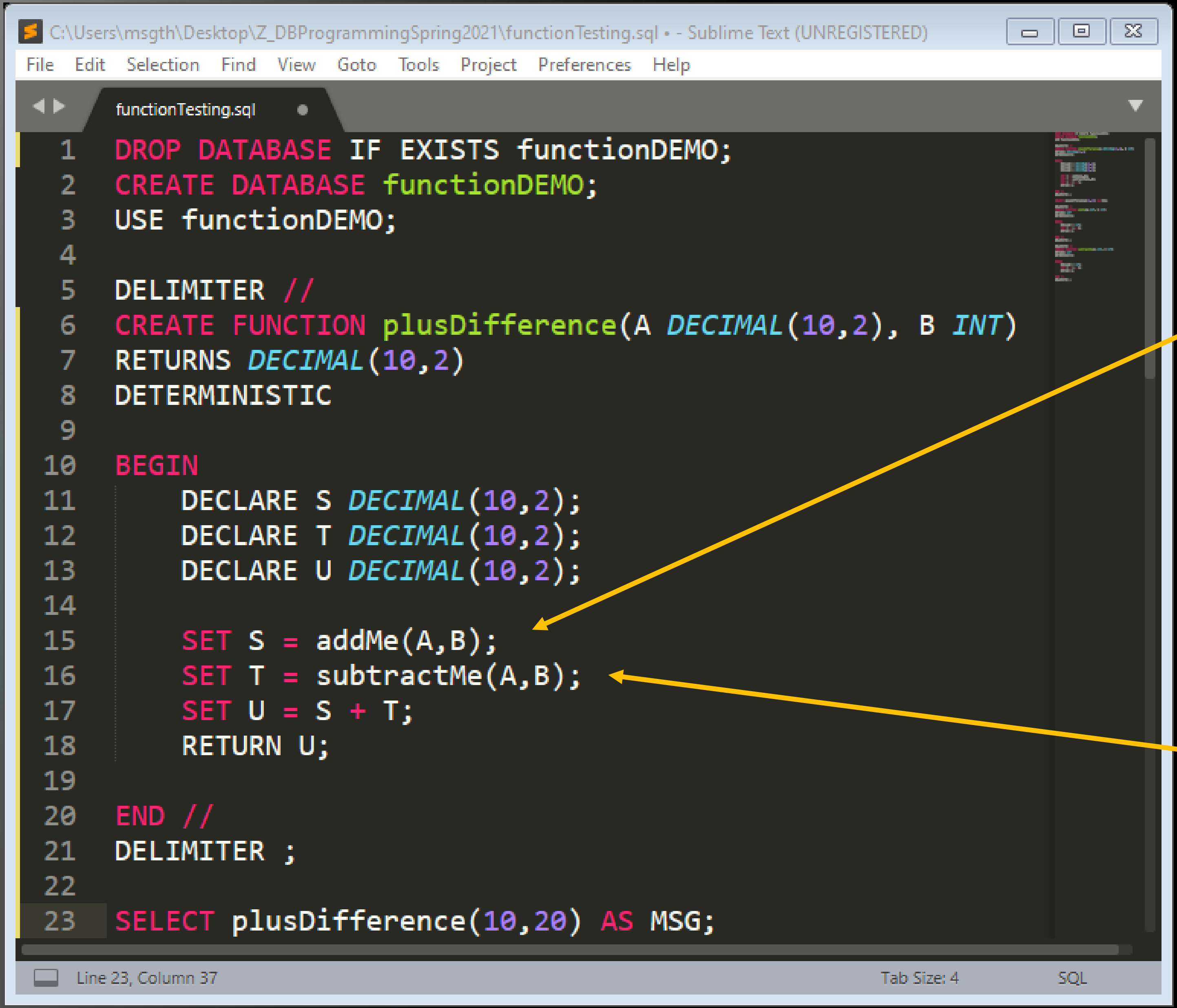
```sql

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

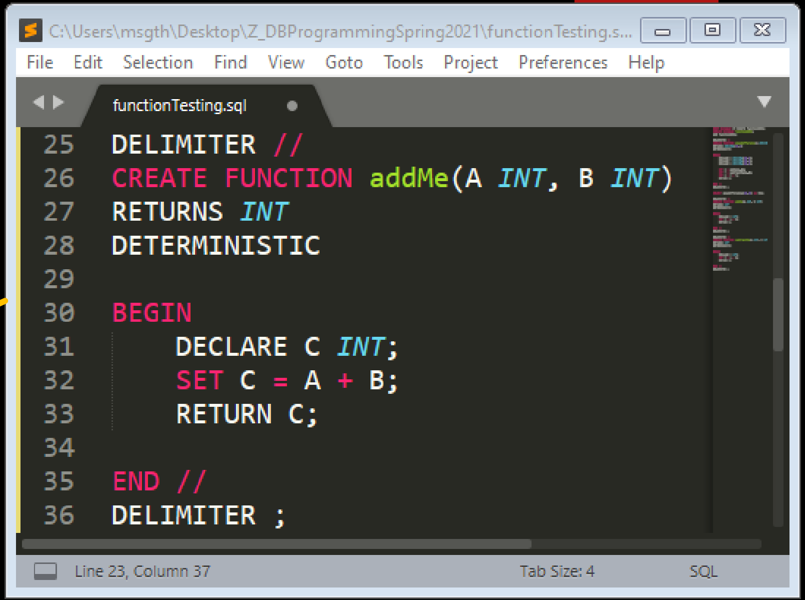
```
+------+
| MSG  |
+------+
| 40   |
+------+
1 row in set (0.00 sec)
```

- 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.

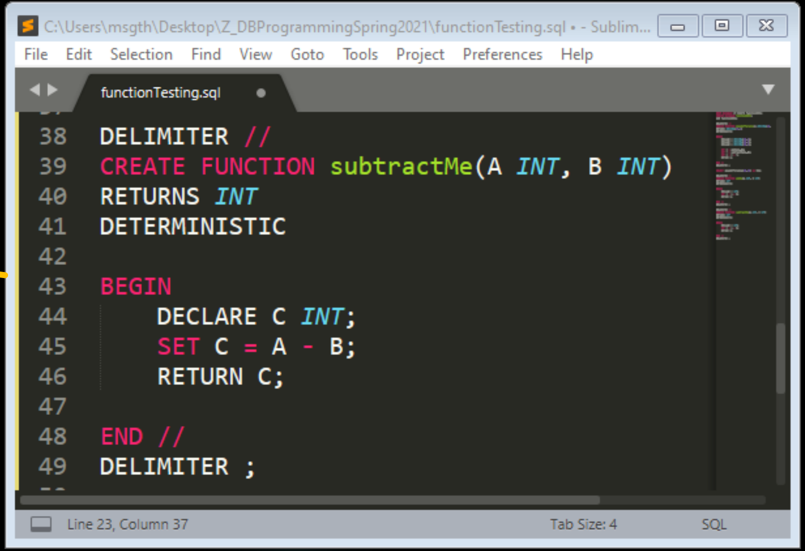
UDF inside UDF


C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
functionTesting.sql
Line 23, Column 37
Tab Size: 4
SQL

```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;
```

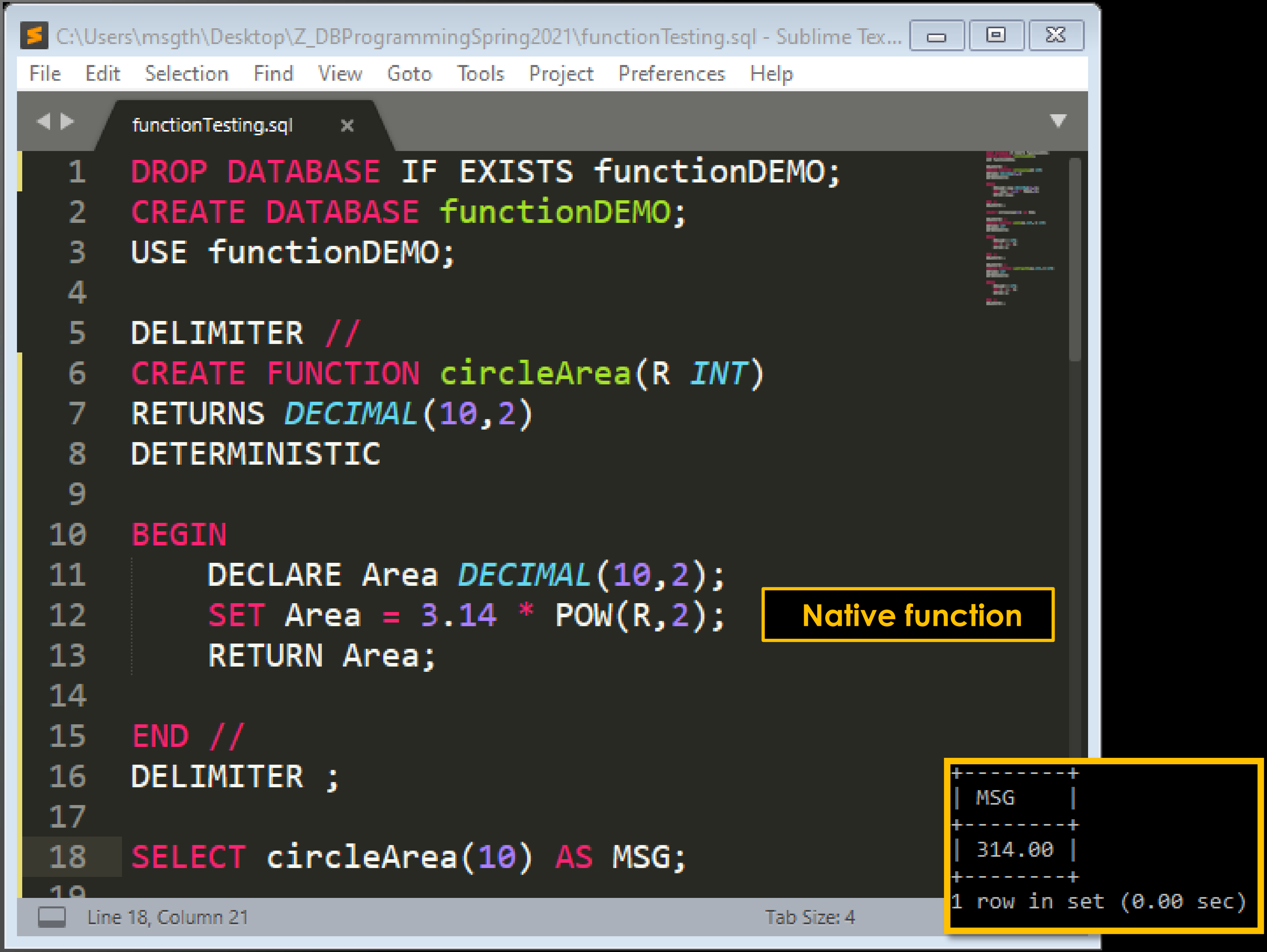

C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.s...
File Edit Selection Find View Goto Tools Project Preferences Help
functionTesting.sql
Line 23, Column 37
Tab Size: 4
SQL

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

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

END //
DELIMITER ;
```


C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql • - Sublim...
File Edit Selection Find View Goto Tools Project Preferences Help
functionTesting.sql
Line 23, Column 37
Tab Size: 4
SQL

```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



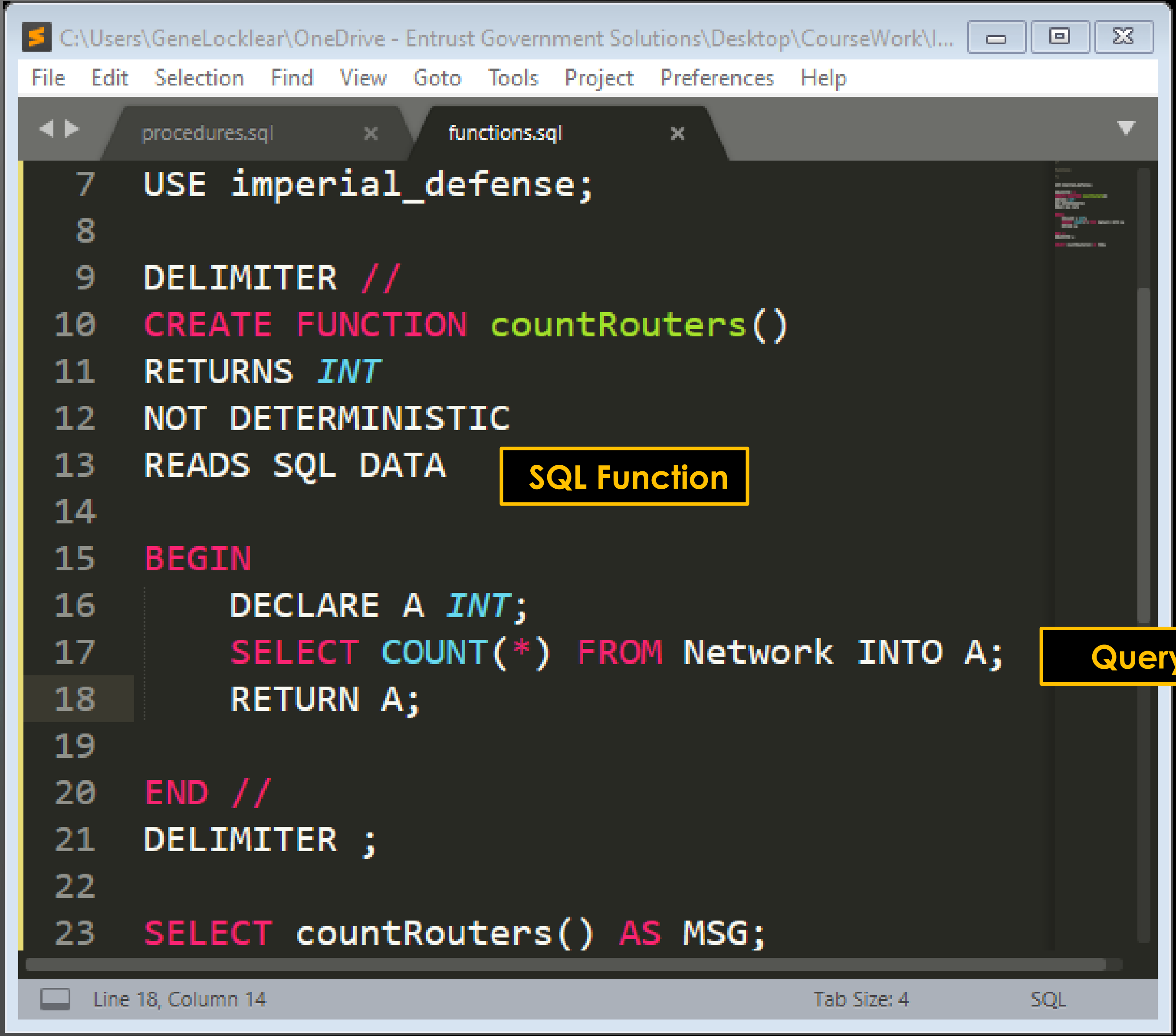
C:\Users\msgth\Desktop\Z_DBProgrammingSpring2021\functionTesting.sql - Sublime Tex...
File Edit Selection Find View Goto Tools Project Preferences Help
functionTesting.sql
Line 18, Column 21
Tab Size: 4

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

DELIMITER //
CREATE FUNCTION circleArea(R INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC

BEGIN
    DECLARE Area DECIMAL(10,2);
    SET Area = 3.14 * POW(R,2);
    RETURN Area;

END //
DELIMITER ;

SELECT circleArea(10) AS MSG;
```

Native function

```
+--------+
| MSG    |
+--------+
| 314.00 |
+--------+
1 row in set (0.00 sec)
```

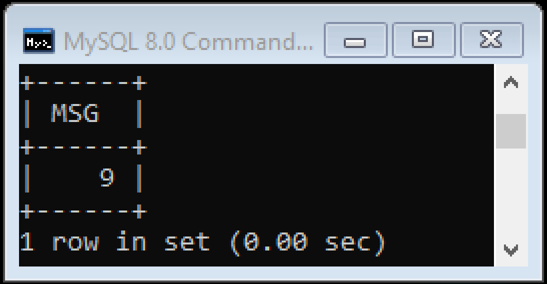
SQL and User-Defined Functions


C:\Users\GeneLocklear\OneDrive - Entrust Government Solutions\Desktop\CourseWork\I...
File Edit Selection Find View Goto Tools Project Preferences Help
procedures.sql
functions.sql
Line 18, Column 14
Tab Size: 4
SQL

```sql
USE imperial_defense;

DELIMITER //
CREATE FUNCTION countRouters()
RETURNS INT
NOT DETERMINISTIC
READS SQL DATA

BEGIN
    DECLARE A INT;
    SELECT COUNT(*) FROM Network INTO A;
    RETURN A;

END //
DELIMITER ;

SELECT countRouters() AS MSG;
```

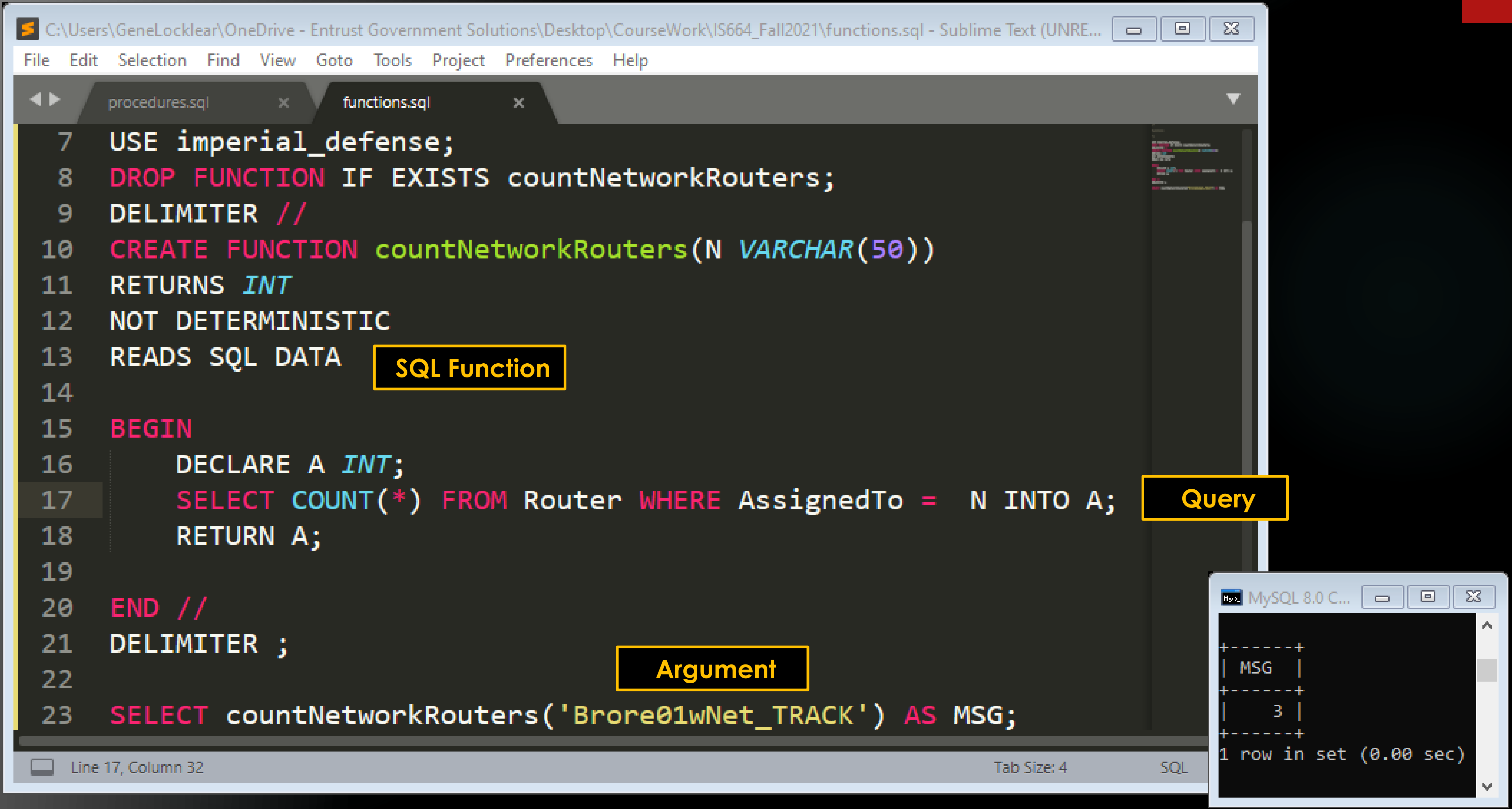
SQL Function

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
Line 17, Column 32
Tab Size: 4
SQL
MySQL 8.0 C...

```sql
USE imperial_defense;
DROP FUNCTION IF EXISTS countNetworkRouters;
DELIMITER //
CREATE FUNCTION countNetworkRouters(N VARCHAR(50))
RETURNS INT
NOT DETERMINISTIC
READS SQL DATA

BEGIN
    DECLARE A INT;
    SELECT COUNT(*) FROM Router WHERE AssignedTo =  N INTO A;
    RETURN A;

END //
DELIMITER ;

SELECT countNetworkRouters('Brore01wNet_TRACK') AS MSG;
```

SQL Function

Query

Argument

```
+------+
| 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