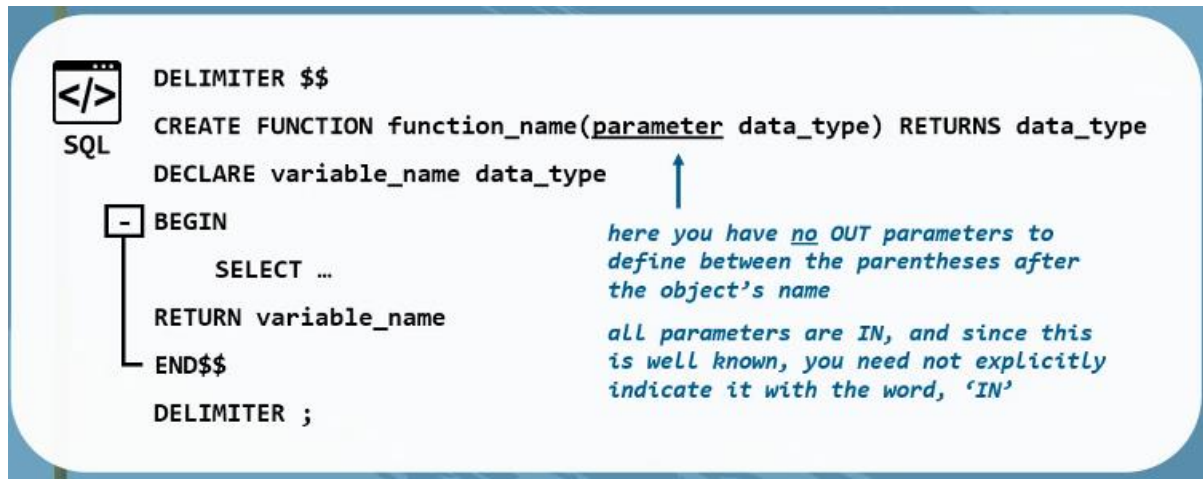


The Difference Between Stored Procedures & Functions:

What a function's syntax looks like:

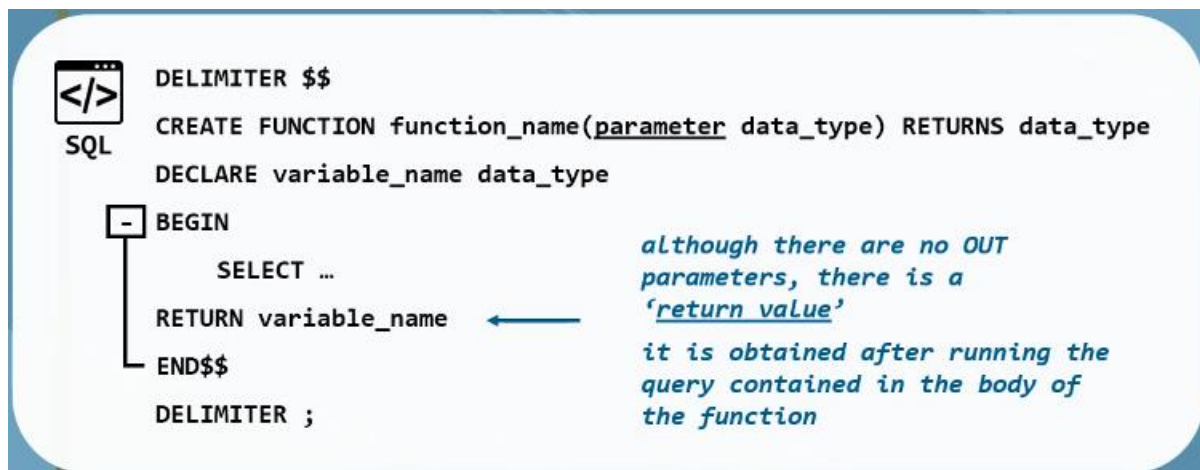


```
SQL
DELIMITER $$
CREATE FUNCTION function_name(parameter data_type) RETURNS data_type
DECLARE variable_name data_type
BEGIN
    SELECT ...
    RETURN variable_name
END$$
DELIMITER ;
```

Annotations:

- ↑ *here you have no OUT parameters to define between the parentheses after the object's name*
- all parameters are IN, and since this is well known, you need not explicitly indicate it with the word, 'IN'*

Writing the parameter name and its data type is enough.



```
SQL
DELIMITER $$
CREATE FUNCTION function_name(parameter data_type) RETURNS data_type
DECLARE variable_name data_type
BEGIN
    SELECT ...
    RETURN variable_name
END$$
DELIMITER ;
```

Annotations:

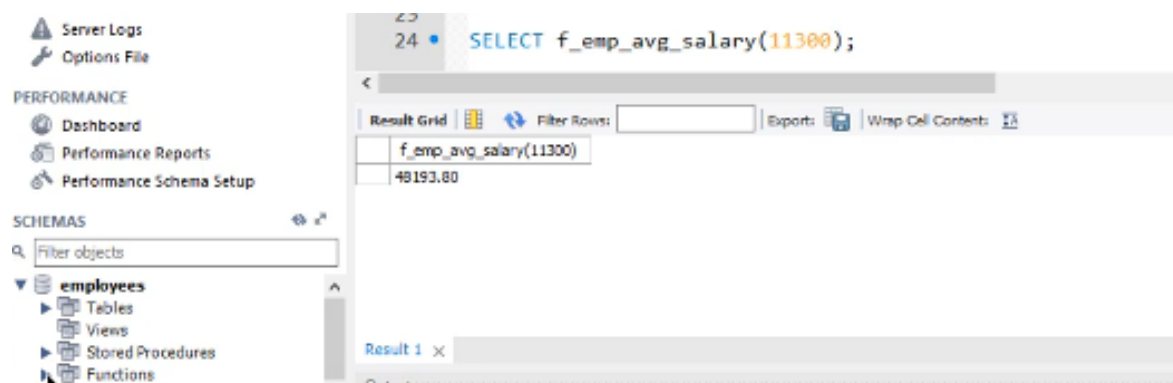
- ← *although there are no OUT parameters, there is a 'return value'*
- it is obtained after running the query contained in the body of the function*

It can be of any data type, so the approach is almost identical to the one used when creating procedures. That's why the code structure is similar, too.

What it would look like:

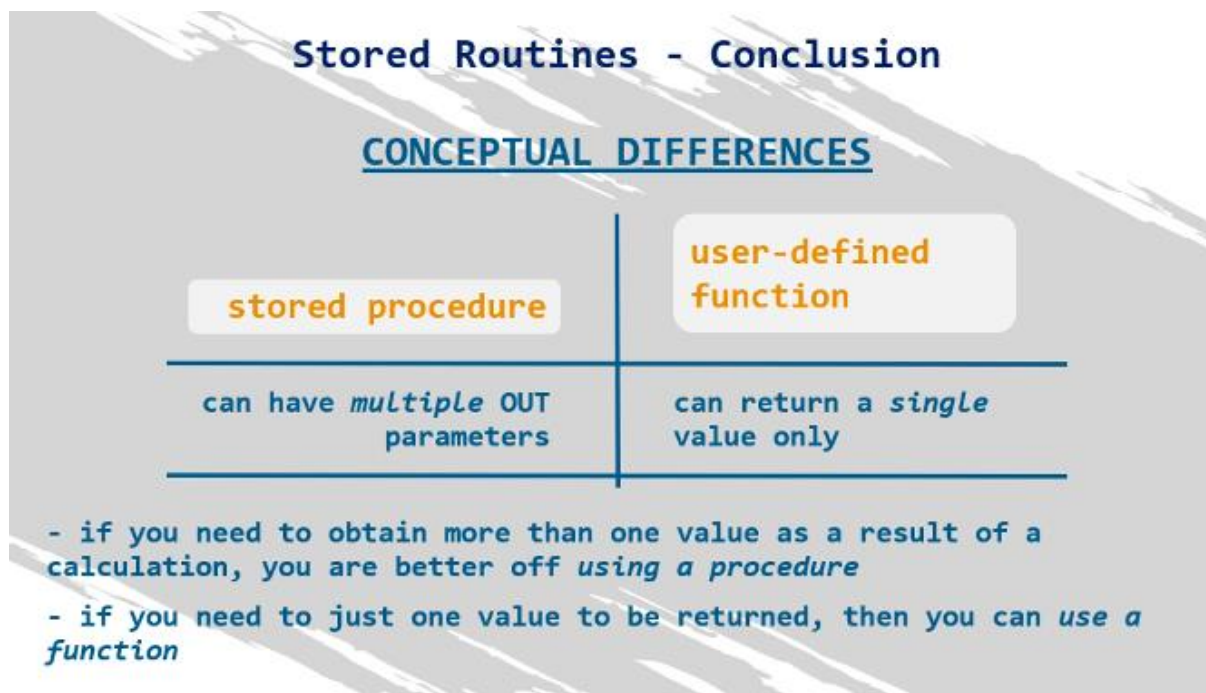
```
1 • USE employees;
2 • DROP function IF EXISTS f_emp_avg_salary;
3
4 DELIMITER $$
5 • CREATE FUNCTION f_emp_avg_salary (p_emp_no INTEGER) RETURNS DECIMAL(10,2)
6 BEGIN
7
8 DECLARE v_avg_salary DECIMAL(10,2);
9
10 SELECT
11     AVG(s.salary)
12 INTO v_avg_salary FROM
13     employees e
14     JOIN
15     salaries s ON e.emp_no = s.emp_no
16 WHERE
17     e.emp_no = p_emp_no;
18
19 RETURN v_avg_salary;
20 END$$
```

Then, instead of designating an OUT parameter, the keyword RETURNS must be written outside the parentheses. After that, we should not indicate an object name, but a data type instead, e.g. DECIMAL. We're still talking about a return value and not a variable. The new variable will be created on the next row, not with the set keyword, but with DECLARE, because this is the word used to create variables visible to the body of the object they belong to. Then, we must indicate the name and the data type of the variable: V_average salary, where V stands for variable. The data type used in the variable line must coincide with the one specified in the create function statement, so DECIMAL (10,2). Lastly, we'll have to insert a return statement, which merely returns the V average salary value. If we omit this line from the function's body, MySQL will display an error because, conceptually, we would not have satisfied the requirement to set a return value when creating a function.



Also, we can't call a function. We can select it, indicating an input value within parentheses.

When we now execute this little query, we will obtain the well-known output of approximately \$48,000. Or we can just run it from the Schemas sidebar.



Stored Routines - Conclusion

how about involving an INSERT, an UPDATE, or a DELETE statement?

- in those cases, the operation performed will apply changes to the data in your database
- there will be no value, or values, to be returned and displayed to the user

Stored Routines - Conclusion

CONCEPTUAL DIFFERENCES

stored procedure	user-defined function
can have <i>multiple</i> OUT parameters	can return a <i>single</i> value only
INSERT UPDATE DELETE	INSERT UPDATE DELETE

The third substantial difference between procedures and user-defined functions regards the way they can be called in a select statement. This refers to a technical distinction we discussed earlier in the section, which was mentioned at the beginning of this lecture as well. Procedures are invoked using the call keyword, whereas functions are referenced in a select statement. What this means is that you can easily include a function as one of the columns inside a select statement. For example, we can include our function calculating the average employee's salary after the employee's last name. Of course, we can use an alias to rename the column to average salary, then our query would look like this.

```
1 • SET @v_emp_no = 11300;  
2 • SELECT  
3     emp_no,  
4     first_name,  
5     last_name,  
6     f_emp_avg_salary(@v_emp_no) AS avg_salary  
7 FROM  
8     employees  
9 WHERE  
10    emp_no = @v_emp_no;
```

We need to remember that including a procedure in a select statement is impossible. Once we've become advanced SQL users and have gained a lot of practice, we will appreciate the advantages and disadvantages of both types of programs.