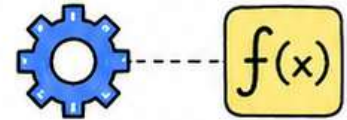


Chapter 20: MySQL Stored Functions

20.1 User-Defined Functions

- * A user-defined function is a set of SQL statements that performs a **specific task** and returns a **single value**.
- * Functions make the code **reusable**, easier to read and **reduce repetition**.
- * Functions can take **input parameters**, process the data and return a **result**.
- * User-defined functions are **stored** in the database and can be used in SQL queries.

```
CREATE FUNCTION  
fn_example(...)  
RETURNS type  
...  
END
```



Key Points :

- ✓ Functions must return a single value.
- ✓ They are created using **CREATE FUNCTION** statement.
- ✓ They improve code reusability and maintainability.
- ✓ They can be used in **SELECT, WHERE, HAVING** and other clauses.



20.2 CREATE FUNCTION

- * The **CREATE FUNCTION** statement is used to create a new user-defined function.
- * It defines the **name** of the function, **input parameters**, **return type** and the SQL statements that form the **function body**.

Syntax :

```
CREATE FUNCTION function_name (param1 datatype, param2 datatype, ...)  
RETURNS return_datatype  
BEGIN  
    -- SQL statements  
    RETURN value;  
END ;
```



Parameters :

Parameter	Description
function_name	Name of the function.
param1, param2, ...	Input parameters (optional).
return_datatype	Data type of the value returned.
BEGIN ... END	Contains the SQL statements of the function.
RETURN value	Returns a single value from the function.

Functions
always return
a **single value**.

Example :

```
DELIMITER $$  
CREATE FUNCTION get_sum(a INT, b INT)  
RETURNS INT  
BEGIN  
    DECLARE result INT;  
    SET result = a + b;  
    RETURN result;  
END $$  
DELIMITER ;
```

Explanation :

- * Function name : get_sum
- * Takes two integers a and b.
- * Adds them and returns the result.
- * Return type : INT
- * Can be called in SQL queries.



Using the Function :

```
SELECT get_sum(10, 20) AS Total;  
-- Output : 30
```

20.3 Function Parameters

- * Parameters are the values passed to a function when it is called.
- * They allow the function to work with different data.
- * MySQL functions can have **zero or more parameters**.
- * Parameters are specified in parentheses in the **CREATE FUNCTION** statement and when calling the function.



Syntax :

```
CREATE FUNCTION function_name (param1 datatype, param2 datatype, ...)
RETURNS return_datatype
BEGIN
    -- SQL statements
    RETURN value;
END ;
```

Types of Parameters :

Type	Description	Example
IN (Default)	Used to pass values to the function.	IN a INT, IN b INT
OUT	Used to return values from the function through parameters.	OUT result INT
INOUT	Used to pass a value to the function and also return a value.	INOUT value INT

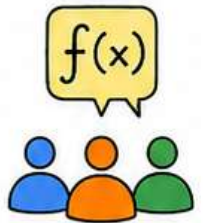
IN is optional.
By default,
parameters are
IN parameters.

Example :

```
DELIMITER $$
CREATE FUNCTION add_numbers (a INT, b INT)
RETURNS INT
BEGIN
    DECLARE sum INT;
    SET sum = a + b;
    RETURN sum;
END $$
DELIMITER ;
```

Explanation :

- * The function add_numbers takes two parameters a and b.
- * It adds them and returns the result.
- * Both parameters are IN by default.

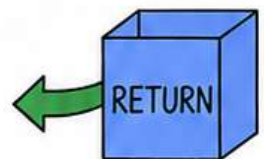


Using the Function :

```
SELECT add_numbers(10, 20) AS Total;
-- Output : 30
```

20.4 RETURN

- * The **RETURN** statement is used to send a value back from the function.
- * A function must return a **single** value of the specified data type.
- * Once **RETURN** is executed, the function ends.
- * The value returned can be used in SQL queries and expressions.



Syntax :

```
RETURN value;
```

Example :

```
DELIMITER $$
CREATE FUNCTION square(n INT)
RETURNS INT
BEGIN
    RETURN n * n;
END $$
DELIMITER ;
```

Explanation :

- * The function square returns the square of the number n.
- * The **RETURN** statement sends the result back to the caller.



Using the Function :

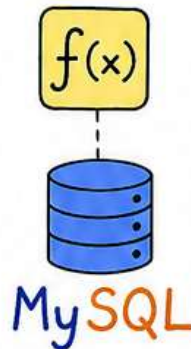
```
SELECT square(5) AS Square_Value;
-- Output : 25
```



Parameters allow functions to work with different inputs, while **RETURN** sends a single value back to be used in SQL.

20.5 Calling Stored Functions

- * Once a function is created, it can be called just like a **built-in function** in SQL.
- * Functions can be used in **SELECT, WHERE, INSERT, UPDATE, DELETE, ORDER BY, HAVING** and other clauses.
- * The **function** is called using its **name** followed by parentheses with **arguments** (if any).



Syntax :

```
SELECT function_name(arg1, arg2, ...);  
-- or use in expressions / clauses
```

Example 1 :

```
-- Calling a function in SELECT  
SELECT get_sum(10, 20) AS Total;
```

Output :

Total
30

Example 2 :

```
SELECT name, get_sum(salary, bonus) AS Total_Amount  
FROM employees;
```

Functions can be used anywhere an **expression** is allowed.

Common Uses :

Clause	Example	Description
SELECT	SELECT get_sum(10,20);	Call function in select list.
WHERE	SELECT * FROM emp WHERE get_sum(sal, bonus) > 50000;	Use in where condition.
ORDER BY	SELECT * FROM emp ORDER BY get_sum(sal, bonus) DESC;	Use for sorting.
HAVING	SELECT dept, get_sum(sal, bonus) AS total FROM emp GROUP BY dept HAVING total > 100000;	Use in having clause.



20.6 DROP FUNCTION

- * The **DROP FUNCTION** statement is used to remove an existing user-defined function from the database.
- * After dropping, the function can no longer be used.
- * Use **DROP FUNCTION** carefully as it **cannot be undone**.



Syntax :

```
DROP FUNCTION function_name;
```

Example :

```
DROP FUNCTION get_sum;
```

Explanation :

- * Deletes the function named get_sum.
- * After this, calling get_sum() will result in an error.



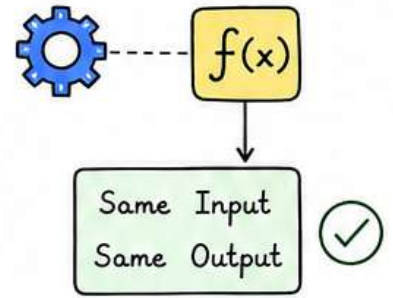
Key Takeaway :



Functions improve code reusability. Use **CREATE FUNCTION** to create, call them in queries and **DROP FUNCTION** when they are no longer needed.

20.7 Deterministic Functions

- * Deterministic functions always return the **same result** for the **same input values**.
- * They **do not depend** on database state, random values, or external data.
- * By default, MySQL assumes functions are **deterministic** unless otherwise specified.
- * Deterministic functions are **safe** to use in statements like indexing, replication and caching.



Syntax :

```
CREATE FUNCTION function_name (param1 datatype, param2 datatype, ...)
RETURNS return_datatype
DETERMINISTIC
BEGIN
    -- SQL statements
    RETURN value;
END ;
```



Example :

```
DELIMITER $$
CREATE FUNCTION square(n INT)
RETURNS INT
DETERMINISTIC
BEGIN
    RETURN n * n;
END $$
DELIMITER ;
```

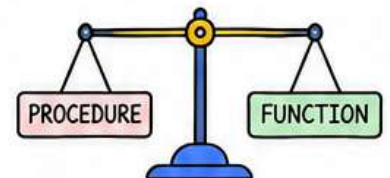
Explanation :

- * The function square() always returns the same result for the same input n.
- * It does not depend on any external data or database state.
- * Hence, it is declared **DETERMINISTIC**.



20.8 Procedure vs Function

- * Both procedures and functions are stored programs in MySQL, but they have some **key differences**.



Feature	Procedure	Function
Returns	Does not return a value directly.	Returns a single value .
Used In	Used with CALL statement.	Used in SELECT, WHERE, HAVING, INSERT, UPDATE , etc.
Purpose	Used to perform operations.	Used to get a result.
Parameters	Can have IN, OUT, INOUT parameters.	Can have IN parameters only.
Return Type	No return type.	Must have a return type.
Example	CALL get_employee();	SELECT get_sum(10, 20);
Use Case	Complex tasks with multiple steps and no direct result.	Calculations and returning a value for use in queries.

Key Takeaway :

- ✓ Use **PROCEDURE** when you want to perform actions.
- ✓ Use **FUNCTION** when you want to return a value.
- ✓ Functions can be used in SQL expressions, procedures cannot.

