

## MySQL CREATE PROCEDURE statement

This query returns all products in the `products` table from the sample database.

```
SELECT * FROM products;
```

The following statement creates a new stored procedure that wraps the query:

```
DELIMITER //

CREATE PROCEDURE GetAllProducts()
BEGIN
    SELECT * FROM products;
END //

DELIMITER ;
```

To execute these statements:

First, launch phpMyAdmin

Select your database and goto SQL section

Enter the statements in the SQL tab:

```
1  DELIMITER //
2
3  • CREATE PROCEDURE GetAllProducts()
4  BEGIN
5      SELECT * FROM products;
6  END //
7
8  DELIMITER ;
```

execute the statements. Note that you can select all statements in the SQL tab (or nothing) and click the **Execute** button. If everything is fine, MySQL will create the stored procedure and save it in the server

In this syntax

- First, specify the name of the stored procedure that you want to create after the CREATE PROCEDURE keywords.
- Second, specify a list of comma-separated parameters for the stored procedure in parentheses after the procedure name. Note that you'll learn how to create stored procedures with parameters in the upcoming tutorials.
- Third, write the code between the BEGIN END block. The above example just has a simple SELECT statement. After the END keyword, you place the delimiter character to end the procedure statement.

## Executing a stored procedure

To execute a stored procedure, you use the CALL statement:

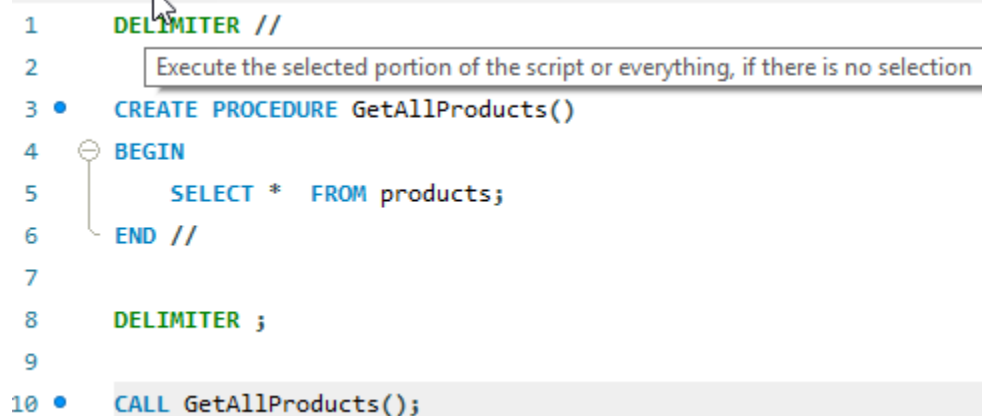
```
CALL stored_procedure_name(argument_list);
```

In this syntax, you specify the name of the stored procedure after the CALL keyword. If the stored procedure has parameters, you need to pass arguments inside parentheses following the stored procedure name.

This example illustrates how to call the GetAllProducts() stored procedure:

```
CALL GetAllProducts();
```

Executing this statement is the same as executing an SQL statement:



```
1 DELIMITER //
2
3 • CREATE PROCEDURE GetAllProducts()
4   BEGIN
5     SELECT * FROM products;
6   END //
7
8 DELIMITER ;
9
10 • CALL GetAllProducts();
```

Execute the selected portion of the script or everything, if there is no selection

*Here's the partial output:*

	productCode	productName	productLine	productScale	productVendor
▶	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast
	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations
	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics
	S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast
	S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics
	S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast
	S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design
	S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast

## Parametrized Store Procedure:

- IN : Identifies the parameter as an input parameter to the procedure.
- OUT : Identifies the parameter as an output parameter that is returned by the procedure.
- INOUT : Identifies the parameter as both an input and output parameter for the procedure.
- datatype : Specifies the data type of the parameter(s).

### *Example of an INOUT parameter*

Suppose we want to get the total count of courses based on the rating. The input parameter is `param_rating` in the procedure, and the data type is **varchar(10)**. The output parameter is ***courses\_count***, and the data type is an **integer**.

Example:

```

16 CREATE PROCEDURE sp_CountMoviesByRating_Inout(
17     INOUT Movies_count INT,
18     IN param_rating VARCHAR(10)
19 )
20 BEGIN
21     SELECT
22         COUNT(title)
23     INTO Movies_count
24 FROM
25     film
26 WHERE
27     rating = param_rating ;
28 END //
29 DELIMITER
30 ;

```

## *Creating parameterized procedure*

```

1  DELIMITER
2  //
3  CREATE PROCEDURE all_courses(IN course_id INT(11))
4  BEGIN
5      SELECT
6          *
7      FROM
8          services
9      WHERE
10         id = course_id ;
11 END //
12 DELIMITER
13 ;

```

To execute the procedure, use the call keyword to call the procedure;

```
Call sp_getCustomers
```

To drop the procedure you can simply use the below statement:

```
Drop procedure sp_getCustomers
```

### Exercise:

#### Task:1

- Create sample stored procedures to insert, update n delete the records with parameter's check the usage of output parameter in the stored procedure.
- Try default or optional parameters and use them inside if there is a value passed  
Call one sp from another sp.