**Summary**: in this tutorial, you will learn how to develop stored procedures that return multiple values.

[MySQL stored function](http://www.mysqltutorial.org/mysql-stored-function/) returns only one value. To develop stored programs that return multiple values, you need to use [stored procedures](http://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx) with INOUT or OUT parameters.

If you are not familiar with INOUT or OUT parameters, check it out the [stored procedure’s parameters tutorial](http://www.mysqltutorial.org/stored-procedures-parameters.aspx) for the detailed information.

Stored procedures that return multiple values example

Let’s take a look at the orders table in the [sample database](http://www.mysqltutorial.org/mysql-sample-database.aspx).

orders table

The following stored procedure accepts customer number and returns the total number of orders that was shipped, canceled, resolved, and disputed.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | DELIMITER $$    CREATE PROCEDURE get\_order\_by\_cust(  IN cust\_no INT,  OUT shipped INT,  OUT canceled INT,  OUT resolved INT,  OUT disputed INT)  BEGIN  -- shipped  SELECT              count(\*) INTO shipped          FROM              orders          WHERE              customerNumber = cust\_no                  AND status = 'Shipped';    -- canceled  SELECT              count(\*) INTO canceled          FROM              orders          WHERE              customerNumber = cust\_no                  AND status = 'Canceled';    -- resolved  SELECT              count(\*) INTO resolved          FROM              orders          WHERE              customerNumber = cust\_no                  AND status = 'Resolved';    -- disputed  SELECT              count(\*) INTO disputed          FROM              orders          WHERE              customerNumber = cust\_no                  AND status = 'Disputed';    END |

In addition to the IN parameter, the stored procedure takes 4 additional OUT parameters: shipped, canceled, resolved, and disputed. Inside the stored procedure, you use a [SELECT statement](http://www.mysqltutorial.org/mysql-select-statement-query-data.aspx)with the [COUNT function](http://www.mysqltutorial.org/mysql-count/) to get the corresponding total of orders based on the order’s status and assign it to the respective parameter.

To use the get\_order\_by\_cust stored procedure, you pass customer number and four user-defined variables to get the out values.

After executing the stored procedure, you use the SELECT statement to output the variable values.



|  |  |
| --- | --- |
| 1  2 | CALL get\_order\_by\_cust(141,@shipped,@canceled,@resolved,@disputed);  SELECT @shipped,@canceled,@resolved,@disputed; |

MySQL Stored Procedures That Return Multiple Values

Calling stored procedures that return multiple values from PHP

The following code snippet shows you how to call the a stored procedure that returns multiple values from PHP.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | <?php  /\*\*  \* Call stored procedure that return multiple values  \* @param $customerNumber  \*/  function call\_sp($customerNumber)  {      try {          $pdo = new PDO("mysql:host=localhost;dbname=classicmodels", 'root', '');            // execute the stored procedure          $sql = 'CALL get\_order\_by\_cust(:no,@shipped,@canceled,@resolved,@disputed)';          $stmt = $pdo->prepare($sql);            $stmt->bindParam(':no', $customerNumber, PDO::PARAM\_INT);          $stmt->execute();          $stmt->closeCursor();            // execute the second query to get values from OUT parameter          $r = $pdo->query("SELECT @shipped,@canceled,@resolved,@disputed")                    ->fetch(PDO::FETCH\_ASSOC);          if ($r) {              printf('Shipped: %d, Canceled: %d, Resolved: %d, Disputed: %d',                  $r['@shipped'],                  $r['@canceled'],                  $r['@resolved'],                  $r['@disputed']);          }      } catch (PDOException $pe) {          die("Error occurred:" . $pe->getMessage());      }  }    call\_sp(141); |

The user-defined variables, which are preceded by the @ sign, are associated with the database connection, therefore, they are available for accessing between the calls.

In this tutorial, we have shown you how to develop a stored procedure that returns multiple values and how to call it from PHP.