**What is a Stored Procedure?**

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

The following [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement returns all rows in the table customers from the [sample database](https://www.mysqltutorial.org/mysql-sample-database.aspx):

|  |  |
| --- | --- |
|  | SELECT      customerName,      city,      state,      postalCode,      country  FROM      customers  ORDER BY customerName; |

The following [CREATE PROCEDURE](https://www.mysqltutorial.org/getting-started-with-mysql-stored-procedures.aspx) statement creates a new stored procedure that wraps the query above:

|  |
| --- |
| DELIMITER $$    CREATE PROCEDURE disp()  BEGIN      SELECT          customerName,          city,          state,          postalCode,          country      FROM          customers      ORDER BY customerName;  END$$  DELIMITER ; |

call disp()

There are four different types of MySQL procedures:

**1. Procedure with no parameters:**  
A procedure without parameters does not take any input or casts an output indirectly. It is simply called with its procedure name followed by () (without any parameters). It is used for simple queries.

**Example:**  
Consider two tables author and book:

create table author (author\_id integer primary key,

authorName varchar(30),

email varchar (25), gender varchar (6));

create table book (BookId integer not null unique,

ISBN integer primary key,

book\_name varchar (30) not null,

author integer, ed\_num integer,

price integer, pages integer,

foreign key (author) references author (author\_id) on delete cascade);

Inserting values into them:

insert into author values

(1, "Kraig Muller", "Wordnewton@gmail.com", "Male");

insert into author values

(2, "Karrie Nicolette", "karrie23@gmail.com", "Female");

insert into book values

(1, 001, "Glimpses of the past", 1, 1, 650, 396);

insert into book values

(2, 002, "Beyond The Horizons of Venus", 1, 1, 650, 396);

insert into book values

(3, 003, "Ultrasonic Aquaculture", 2, 1, 799, 500);

insert into book values

(4, 004, "Cyrogenic Engines", 2, 1, 499, 330);

Procedure (with no parameters) to display all the books:

delimiter //

create procedure display\_book()

-> begin

-> select \*from book;

-> end //

call display\_book(); //

**2. Procedure with IN parameter:**  
An IN parameter is used to take a parameter as input such as an attribute. When we define an IN parameter in a procedure, the calling program has to pass an argument to the stored procedure. In addition, the value of an IN parameter is protected. It means that even if the value of the IN parameter is changed inside the procedure, its original value is retained after the procedure ends (like pass by value). In other words, the procedure only works on the copy of the IN parameter.

**Example:**  
Procedure to update price of a book taking ISBN of the book and its new price as input: (considering the tables above)

delimiter //

create procedure update\_price (IN temp\_ISBN varchar(10), IN new\_price integer)

-> begin

-> update book set price=new\_price where ISBN=temp\_ISBN;

-> end; //

call update\_price(‘001’, 600); //

We changed the price of book with ISBN ‘001’(Glimpses of the past) to 600 (from its default price 650).

**Output:**

delimiter ;

select \*from book;

+--------+------+------------------------------+--------+--------+-------+-------+

| BookId | ISBN | book\_name | author | ed\_num | price | pages |

+--------+------+------------------------------+--------+--------+-------+-------+

| 1 | 1 | Glimpses of the past | 1 | 1 | 600 | 396 |

| 2 | 2 | Beyond The Horizons of Venus | 1 | 1 | 650 | 396 |

| 3 | 3 | Ultrasonic Aquaculture | 2 | 1 | 799 | 500 |

| 4 | 4 | Cyrogenic Engines | 2 | 1 | 499 | 330 |

+--------+------+------------------------------+--------+--------+-------+-------+

4 rows in set (0.0013 sec)

**3. Procedure with OUT parameter:**  
An OUT parameter is used to pass a parameter as output or display like the select operator, but implicitly (through a set value). The value of an OUT parameter can be changed inside the procedure and its new value is passed back to the calling program. A procedure cannot access the initial value of the OUT parameter when it starts.

**Example:**  
Procedure to display the highest price among all the books with an output parameter:

delimiter //

create procedure disp\_max(OUT highestprice integer)

-> begin

-> select max(price) into highestprice from book;

-> end; //

call disp\_max(@M); //

select @M;

**Output:**  
The highest price from our book database is of the book with ISBN 003 (Ultrasonic Aquaculture) with a price of 799, which is displayed.

+-----+

| @M |

+-----+

| 799 |

+-----+

1 row in set (0.0005 sec)

**4. Procedure with IN-OUT parameter:**  
An INOUT parameter is a combination of IN and OUT parameters. It means that the calling program may pass the argument, and the stored procedure can modify the INOUT parameter and pass the new value back to the calling program.

**Example:**  
Procedure to take gender type input (‘Male’/’Female’ here) with an in-out parameter which reflects the number of authors falling in that gender category/type:

delimiter //

create procedure disp\_gender(INOUT mfgender integer, IN emp\_gender varchar(6))

-> begin

-> select COUNT(gender)

INTO mfgender FROM author where gender = emp\_gender;

-> end; //

delimiter ;

call disp\_gender(@M, "Male");

select @M;

call disp\_gender(@F, "Female");

select @F;

**Output:**  
We have two authors, one being male and one being female as per insertions in the table author. Hence, output is 1 for one male author and 1 for one female author respectively.

+----+

| @M |

+----+

| 1 |

+----+

1 row in set (0.0004 sec)

+----+

| @F |

+----+

| 1 |

+----+

1 row in set (0.0005 sec)

Drop a procedure

In MySQL, a procedure can also be dropped. When a procedure id dropped, it is removed from the database.

Syntax:

1. **Drop** **procedure**[ IF EXISTS ] procedure\_name;